**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФРАСТРУКТУРИ ТА ТЕХНОЛОГІЙ**

ІНСТИТУТ УПРАВЛІННЯ, ТЕХНОЛОГІЙ ТА ПРАВА

Факультет управління і технологій

Кафедра інформаційних технологій та дизайну

«ДО ЗАХИСТУ ДОПУЩЕНО»

Завідувач кафедри інформаційних технологій та дизайну

к.т.н., доц.\_\_\_\_\_\_\_\_\_\_\_\_ В.В. Завгородній

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021 р.

**МАГІСТЕРСЬКА РОБОТА**

(пояснювальна записка)

Тема: *Методологія створення WEB-додатку з використанням React.js*

Спеціальність: *121 «Інженерія програмного забезпечення»*

Розробив:  *Прізвище Ім’я По-батькові* \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ПІБ-повністю) (підпис)

*ІПЗ-20121маг*  *1111111111*

(№ групи для д.ф.н./ курс для з.ф.н.) (№ залікової книжки)

Керівник:  *ПІБ керівника* \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, ПІБ-повністю) (підпис)

Консультанти:\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, ПІБ) (підпис)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, ПІБ) (підпис)

Нормоконтролер:  *ПІБ керівника* \_\_\_\_\_\_\_\_\_\_\_\_\_\_

(посада, вчене звання, науковий ступінь, ПІБ) (підпис)

Київ 2021 р.

**Державний університет інфраструктури та технологій**

Факультет управління і технологій

(повна назва)

Кафедра інформаційних технологій та дизайну

(повна назва)

Рівень вищої освіти другий (магістерський)

(код, назва)

Спеціальність 121 «Інженерія програмного забезпечення»

(код, назва)

ЗАТВЕРДЖУЮ:

Завідувач кафедри інформаційних технологій та дизайну

к.т.н., доц. Завгородній В.В.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_2021 р.

**ЗАВДАННЯ**

на магістерську роботу

*Мартинюку Артему Віталійовичу*

(прізвище, ім’я, по батькові)

1. Тема магістерської роботи *Методологія створення WEB-додатку з використанням React.js*

затверджена наказом по університету від «22» жовтня 2021 р. № 734/01-05/08.1

2. Термін здачі студентом закінченої магістерської роботи «07» грудня 2021 р.

3. Вихідні дані до магістерської роботи *\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Зміст розрахунково-пояснювальної записки (перелік питань, які розробляються)\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Консультанти магістерської роботи, з вказівкою розділів роботи, які до них відносяться

|  |  |  |  |
| --- | --- | --- | --- |
| Розділ | Консультант | Підпис | |
| Завдання видав | Завдання прийняв |
| Огляд та аналіз області дослідження | *ПІБ керівника* |  |  |
| Аналіз та шляхи розв’язку проблеми | *ПІБ керівника* |  |  |
| Опис програмного продукту | *ПІБ керівника* |  |  |

6. Дата видачі завдання «11» жовтня 2021 р.

Керівник \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

Завдання прийняв до виконання \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(підпис)

КАЛЕНДАРНИЙ ПЛАН

|  |  |  |  |
| --- | --- | --- | --- |
| № з/п | Найменування етапів магістерської роботи | Строк виконання етапів магістерської роботи | Примітки |
| 1 | Огляд та аналіз джерел тематичної області | 11.10.21 – 24.10.21 |  |
| 2 | Огляд та аналіз шляхів розв’язання проблем роботи | 25.10.21 – 31.10.21 |  |
| 3 | Опис власного підходу до вирішення проблеми | 01.11.21 – 14.11.21 |  |
| 4 | Визначення вимог до програмного продукту та його структури | 15.11.21 – 21.11.21 |  |
| 5 | Опис та реалізація основних функцій програмного продукту | 22.11.21 – 05.12.21 |  |
| 6 | Оформлення пояснювальної записки | 06.12.21 – 10.12.21 |  |
| 7 | Захист |  |  |

Студент-магістрант *Прізвище І.Б.* (підпис)

Керівник роботи *Прізвище І.Б.* (підпис)

АНОТАЦІЯ

Текст анотації українською мовою Текст анотації українською мовою Текст анотації українською мовою Текст анотації українською мовою Текст анотації українською мовою

АННОТАЦИЯ

Текст аннотации на русском языке Текст аннотации на русском языке Текст аннотации на русском языке Текст аннотации на русском языке Текст аннотации на русском языке

ABSTRACT

Annotation text in English Annotation text in English Annotation text in English Annotation text in English Annotation text in English Annotation text in English Annotation text in English

ЗМІСТ

[ВСТУП 10](#_Toc90598177)

[1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ 12](#_Toc90598178)

[1.1 Про веб-додатки 12](#_Toc90598179)

[1.2 Як працюють веб-додатки 13](#_Toc90598180)

[1.2.1 Обробка статичних веб-сторінок 13](#_Toc90598181)

[1.2.2 Обробка динамічних сторінок 15](#_Toc90598182)

[1.3 Типи веб-додатків 16](#_Toc90598183)

[1.3.1 Що таке SPA-застосунок 17](#_Toc90598184)

[1.3.2 Що таке MPA-застосунок 19](#_Toc90598185)

[1.3.3 Що таке PWA-застосунок 21](#_Toc90598186)

[1.4 Використання фреймворків 21](#_Toc90598187)

[2 АНАЛІЗ ТА ШЛЯХИ РОЗВ’ЯЗКУ ПРОБЛЕМИ 27](#_Toc90598188)

[2.1 React – UI – Фреймворк 27](#_Toc90598189)

[2.1.1 Чому саме React 27](#_Toc90598190)

[2.1.2 React Virtual DOM 37](#_Toc90598191)

[2.1.3 React router dom 39](#_Toc90598192)

[2.2 Redux – Менеджер станів додатку 40](#_Toc90598193)

[2.2.1 FLUX-архітектура 40](#_Toc90598194)

[2.2.2 Redux 45](#_Toc90598195)

[2.2.3 Thunk мидлвар 45](#_Toc90598196)

[2.3 Статична типізація пропсів у React додатку 46](#_Toc90598197)

[2.4 CSS-in-JS 48](#_Toc90598198)

[2.5 Storybook 51](#_Toc90598199)

[2.6 Firebase 52](#_Toc90598200)

[3 ОПИС ПРОГРАМНОГО ПРОДУКТУ 55](#_Toc90598201)

[3.1 3.1 Заголовок першого підрозділу у третьому розділі 55](#_Toc90598202)

[3.2 3.2 Заголовок другого підрозділу у третьому розділі 55](#_Toc90598203)

[ВИСНОВКИ 56](#_Toc90598204)

[СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 57](#_Toc90598205)

Содіржаниє

|  |  |
| --- | --- |
| ВСТУП | 8 |
| 1 ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ |  |
| Про веб-приложение  Про спа  Про пва пргресив |  |
| Про фреймворк |  |
|  |  |
| 2 АНАЛІЗ ТА ШЛЯХИ РОЗВ’ЯЗКУ ПРОБЛЕМИ |  |
|  |  |
| Технологии стек фреймворк реакт |  |
| Роутинг – реакт роутер дом |  |
| Стейт менеджер редакс  Мидлвар санк |  |
| Типизация проп тайпс  Css-in-js  Stories Book  ?Бд файр бейз  ?Авторизация с помощью гугл |  |
| 3 ОПИС ПРОГРАМНОГО ПРОДУКТУ |  |
| Настройка ИДЕ и Проекта  Создание Компонентов  Создание Редакса |  |
|  |  |
|  |  |
| Руководво пользователя |  |
|  |  |
| ВИСНОВКИ |  |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ |  |
| ДОДАТКИ |  |

ВСТУП

JavaScript спочатку був створений як мова сценаріїв для Netscape. Потім він був відправлений в ESMA international для стандартизації. Це призвело до появи новогостандарту ESMAscript. Наступні версії JavaScript вже базувалися на стандарті ESMAscript.

ESMAscript є стандартом, а JavaScript є найпопулярнішою реалізацією цього стандарту.  
 Однак JavaScript все ще був «найбільш незрозумілою мовою програмування в світі». Серед його проблем були глобальні зони видимості змінних, однопоточність, відсутність суворої типізації та принципів ООП.  
 Але в червні 2015 року все змінилося. Був випущений новий стандарт - ES6, додавши наступні зміни:

* деструктивне призначення;
* стрілочні функції;
* шаблонні строки;
* локальні змінні та константи;
* ітератори та генератори;
* класи;
* проміси та async/await.

З тих пір щороку випускається новий стандарт ESMAscript.

*Актуальність.* Про попит на будь-яку мову можна судити по кількості написаних для неї бібліотек. NPM (node package manager) це головний репозиторій пакетів для середы node.js. Він є найбільшим у світі. Станом на 2017 рік в NPM було розміщено понад 380 тисяч пакетів. Для порівняння в репозиторії packagist(php) – 123 тисячі, в PyPI(Python) – 96 тисяч і в nuget (.NET) – 70 тисяч.

В середньому через NPM на місяць встановлюється понад 18 мільярдів пакетів.

Мета дослідження. Метою роботи є дослідження бібліотеки react.js та його можливості для створення клієнтської частини веб-додатку.

Завдання дослідження. Створення робочого зразка програмного забезпеченнях, який демонструє кращі практики створення SPA-додатку.

Об'єкт дослідження. Процес створення додатку за допомогою react.js та redux; порівняння різних методів та підходів створення.

Предмет дослідження. Ретроспектива; web-застосунок, що надає змогу створювати задачі та управляти ними.

1. ОГЛЯД ТА АНАЛІЗ ОБЛАСТІ ДОСЛІДЖЕННЯ
   1. Про веб-додатки

Веб-застосунок – це веб-сайт, на якому розміщено сторінки з частково або повністю несформованим вмістом. Остаточний контент генерується тільки після того, як відвідувач сайту запитує сторінку з веб-сервера. Оскільки кінцевий зміст сторінки залежить від запиту на основі дій відвідувача, сторінка називається динамічною.

На відміну від веб-сайту, веб-застосунок - це повноцінна програма, до якою користувач отримує доступ через інтернет, тобто не вимагає установки на пристрій. Веб-застосунок є інтерактивним і дозволяє користувачам взаємодіяти з різними елементами: наприклад, залишити заявку на покупку товару, зробити покупку авіаквитка або прокоментувати пост друга.

Спектр використання веб-додатків дуже широкий. У цьому розділі розглядаються загальні міркування щодо використання веб-застосунків і наведено приклад простого веб-застосунку.

Типове використання веб-застосунків

Використання веб-додатків приносить певні переваги як відвідувачам веб-сайту, так і їх розробникам.

* Веб-додатки дозволяють відвідувачам швидко і легко знаходити необхідну інформацію на сайтах з великим обсягом інформації.

Цей вид веб-додатків дозволяє шукати контент, впорядковувати контент і переміщатися по ньому зручним для відвідувачів способом. Прикладами таких додатків є внутрішні мережі компаній — Microsoft MSDN ([www.msdn.microsoft.com](http://www.msdn.microsoft.com/)) і Amazon ([www.amazon.com](http://www.amazon.com/)).

* Веб-додатки дозволяють збирати, зберігати і аналізувати дані, отримані від відвідувачів сайту.

Довгий час використовувався метод, в якому введені дані HTML-форми, відправлялись на обробку CGI-додаткам або спеціально призначеним працівникам у вигляді повідомлень електронної пошти. Веб-застосунок дозволяє зберігати дані безпосередньо в базі даних, а також отримувати дані і генерувати звіти на основі даних, отриманих для аналізу. Приклади включають інтерактивні сторінки банків, сторінки для контролю запасів, соціологічні дослідження та опитування, а також форми для зворотного зв'язку користувачів.

* веб-застосунок можна використовувати для оновлення веб-сайтів, які мають вміст, який періодично змінюється.

веб-застосунок звільняє веб-дизайнера від рутинної роботи постійного оновлення html-сторінок сайту. контент-провайдери, такі як редактори новин, несуть відповідальність за наявність свіжого матеріалу, а веб-застосунок стежить за автоматичним оновленням сайту. Прикладом є веб-версія журналу «The Economist» ([www.economist.com](http://www.economist.com/)) и служба новин CNN ([www.cnn.com](http://www.cnn.com/)).

* 1. Як працюють веб-додатки

Будь-який веб-застосунок - це набір статичних і динамічних веб-сторінок. *Статична веб-сторінка* - це сторінка, яка завжди відображається користувачеві в незмінному вигляді. Веб-сервер відправляє сторінку на запит веб-браузера без будь-яких змін. На відміну від цього, сервер вносить зміни на динамічну веб-сторінку перед відправкою її в браузер.

Наприклад, можна створити сторінку, на якій відображаються результати оздоровчої програми. При цьому деяка інформація (наприклад, ім'я співробітника і його результати) буде визначатися в момент запиту сторінки співробітником.

У наступному розділі детально описано, як працюють веб-застосунки.

* + 1. Обробка статичних веб-сторінок

Статичний веб-сайт містить набір відповідних HTML-сторінок і файлів, розміщених на комп'ютері, на якому встановлено веб-сервер.

Веб-сервер - це програмне забезпечення, яке надає веб-сторінки у відповідь на запити веб-браузерів. Зазвичай запит на сторінку створюється натисканням посилання на веб-сторінці, вибором закладки у браузері або введенням URL-адреси в адресному рядку браузера.

Остаточний вміст статичної веб-сторінки визначається розробником і залишається незмінним в процесі запиту сторінки. Приклад:

<html>

<head>

    <title>Trio Motors Information Page</title>

  </head>

  <body>

    <h1>About Trio Motors</h1>

    <p>Trio Motors is a leading automobile manufacturer.</p>

  </body>

</html>

Весь HTML-код генерується розробником до розміщення сторінки на сервері. Оскільки HTML-код не змінюється після розміщення сторінки на сервері, сторінка називається статичною.

Примітка.

Строго кажучи, "статична" сторінки насправді може такою і не бути. Наприклад, зображення заміщення або flash-контент (як файл SWF) дозволяє «оживити» статичну сторінку. Однак у цьому контексті сторінка статична, оскільки вона надсилається до браузера без змін.

Коли веб-сервер отримує запит на видачу статичної сторінки, то, проаналізувавши запит, сервер знаходить потрібну сторінку і відправляє її в браузер, як показано нижче.

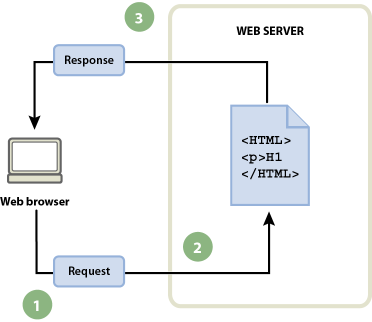


Рисунок 1.3.1 – *Обробка статичної веб-сторінки*

**A.** Веб-браузер запитує статичну сторінку. **B.** Веб-сервер знаходить сторінку. **C.** Веб-сервер надсилає сторінку до браузера, який її запросив.

У випадку веб-додатків деякі частини коду сторінки відсутні, поки відвідувач не запитає сторінку. Відсутній код генерується якимось механізмом, і тільки тоді сторінка може бути відправлена в браузер. Цей механізм генерації коду обговорюється в наступному розділі.

* + 1. Обробка динамічних сторінок

Коли веб-сервер отримує запит на випуск статичної веб-сторінки, він надсилає сторінку безпосередньо до браузера. Однак, коли запитується динамічна сторінка, дії веб-сервера не настільки чіткі. Сервер передає сторінку в спеціальну програму, яка формує кінцеву сторінку. Така програма називається сервером додатків.

Сервер програми зчитує код, який знаходиться на сторінці, генерує кінцеву сторінку відповідно до прочитаного коду, а потім видаляє її зі сторінки. В результаті всіх цих операцій виходить статична сторінка, яка передається на веб-сервер, який в свою чергу відправляє її в клієнтський браузер. Всі сторінки, які отримує браузер, містять тільки HTML-код. Схематичне представлення процесу:

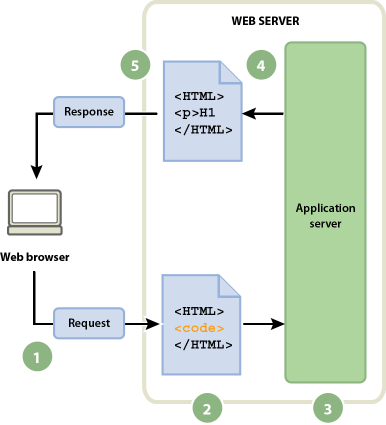


Рисунок 1.3.2 – *Обробка динамічних сторінок*

**A.** Веб-браузер запитує динамічну сторінку. **B.** Веб-сервер знаходить сторінку та передає її серверу застосунків. **C.** Сервер програми сканує сторінку на наявність інструкцій і створює її. **D.** Сервер застосунків повертає підготовлену сторінку на веб-сервер. **E.** Веб-сервер надсилає підготовлену сторінку до браузера, який її запросив.

* 1. Типи веб-додатків

Веб-додатки можна класифікувати по-різному: в залежності від їх функціональності і призначення. Давайте детальніше розглянемо ці типи додатків, щоб краще зрозуміти, як вони працюють і який з них підходить для ваших бізнес-завдань.

Існує три основні шаблони для створення веб-сайтів:

* MPA (multi-page application): багатосторінковий застосунок, який надсилає запит на сервер і повністю оновлює сторінку під час виконання на ній дії.
* SPA (single-page application): односторінковий застосунок, який містить HTML-сторінку, яка динамічно оновлюється на основі дій користувача без повного перезавантаження.
* PWA (progressive web application): застосунок, який користувач встановлює і може використовувати в автономному режимі.
  + 1. Що таке SPA-застосунок

Значення терміна SPA (Single Page Application) лежить в собі. SPA - це односторінковий застосунок, який містить HTML-сторінку, яка динамічно (без повного перезавантаження) оновлюється у відповідь на дії користувача. Архітектура програми розроблена так, що при початковому запуску відвідувач бачить основний зміст сайту в браузері, а нові дані завантажуються в міру необхідності, наприклад, при прокручуванні або натисканні на іконку. Якщо ви коли-небудь перегорнули стрічку Facebook, Twitter або Instagtam, ви знаєте, про що йде мова.

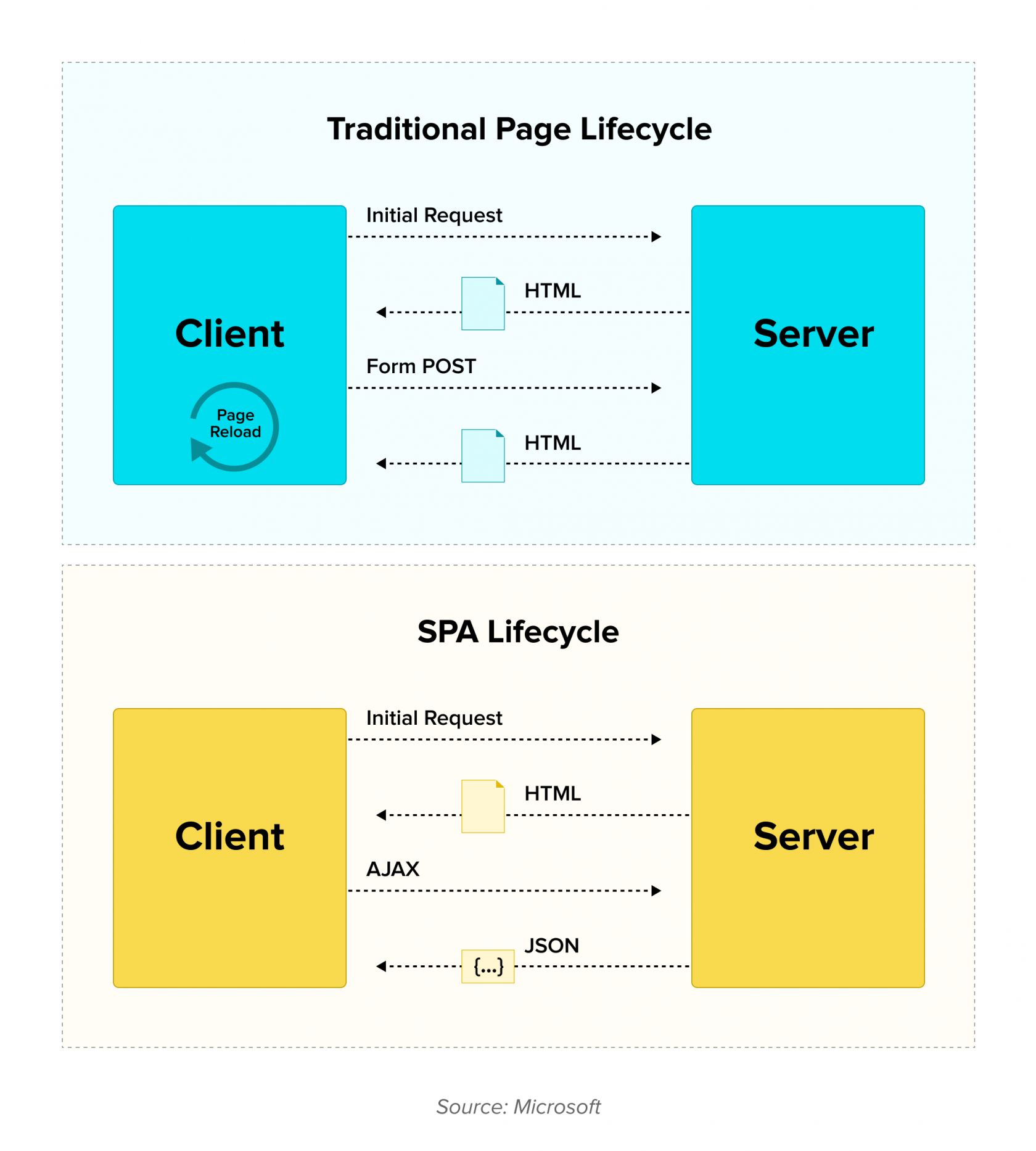


Рисунок 1.4.1 – *Життевий цикл SPA*

Насправді односторінкові додатки багато в чому схожі на настільні: при переході на нову сторінку оновлюється тільки частина контенту, що дозволяє не завантажувати одні і ті ж елементи сайту багато разів. Подібний ефект для SPA забезпечують сучасні javascript фреймворки. Для невеликих проектів підійде jQuery, але для створення серйозного SPA вам знадобляться фронтенд-бібліотеки, такі як React, Vue або Angular (рис 1.4.2).

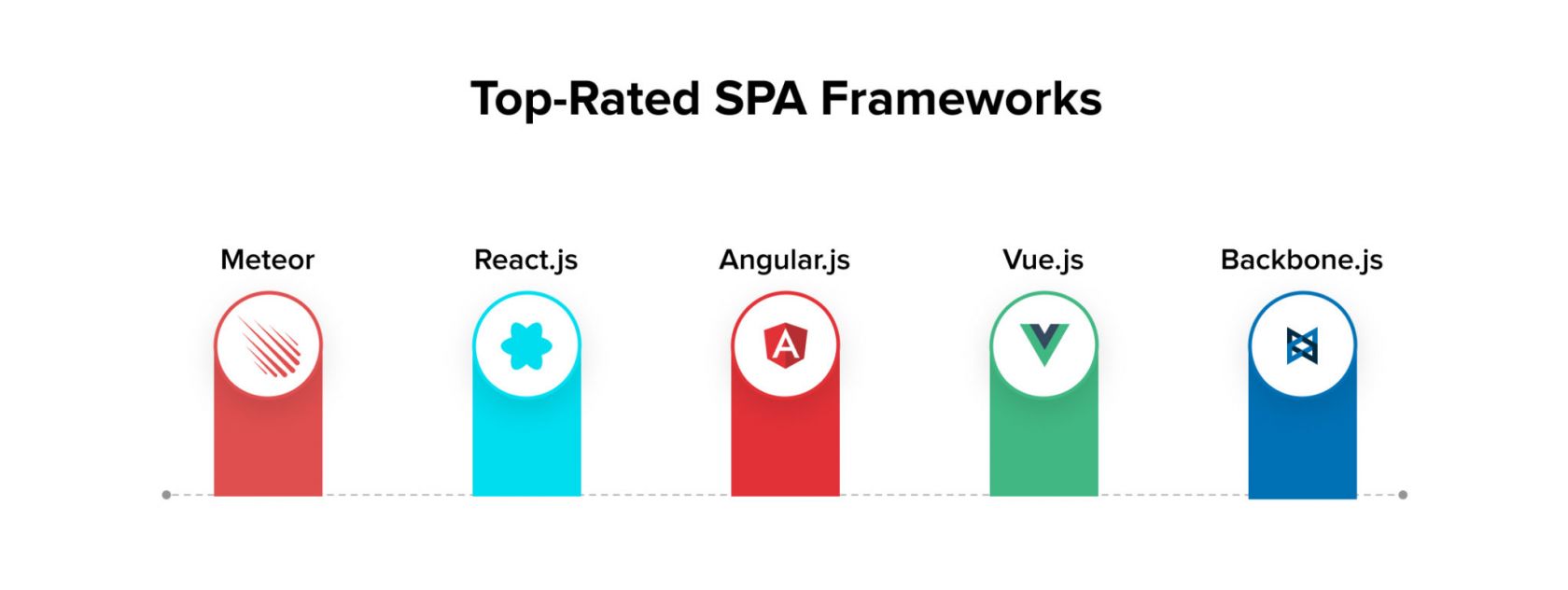


Рисунок 1.4.2 – *Найпопулярніші SPA фреймворки*

SPA-додатки використовуються в тих випадках, коли на сайт мало інформації і вона повинна бути ефективно продемонстрована користувачеві. Прикладами SPA є соціальні мережі та сайти з великою кількістю користувачів.

Переваги SPA

Дослідження показують, що користувачі чутливі до швидкості завантаження сторінок. У питанні швидкості завантаження SPA-сайти не мають рівних, що є їх головною перевагою. Так як левова частка ресурсів завантажується за один сеанс, при подальших діях SPA швидко оновлює необхідну частину даних, значно заощаджуючи час.

Однак висока швидкість - не єдиний плюс SPA-додатків. Крім того, односторінкові додатки мають наступні переваги:

* Легкість створення. Бібліотеки та фреймворки вже готові до розробки SPA-додатків, робота над інтерфейсом та бекендом може проводитися паралельно. Крім того, на основі готового коду в майбутньому можна побудувати мобільний застосунок;
* гнучкість інтерфейсу користувача. Для однієї сторінки простіше розробити захоплюючий та інтерактивний дизайн інтерфейсу;
* Просте кешування даних. Важливі дані завантажуються в одному запиті, і тоді користувач односторінкового додатка може працювати частково в автономному режимі, підключаючись до Інтернету для збереження прогресу.
  + 1. Що таке MPA-застосунок

MPA або Multi Page Applications за принципом роботи є зовсім протилежний SPA. MPA - це багатосторінковий застосунок, який працює як веб-сайт, до якого ми звикли. Вони відправляють запит на сервер і повністю оновлюють сторінку при виконанні з нею будь-якої дії (переходячи на іншу сторінку, вводячи і змінюючи дані). Ця архітектура додатків має значний вплив на швидкість і продуктивність, оскільки більшість даних перезавантажується на кожному стрибку.

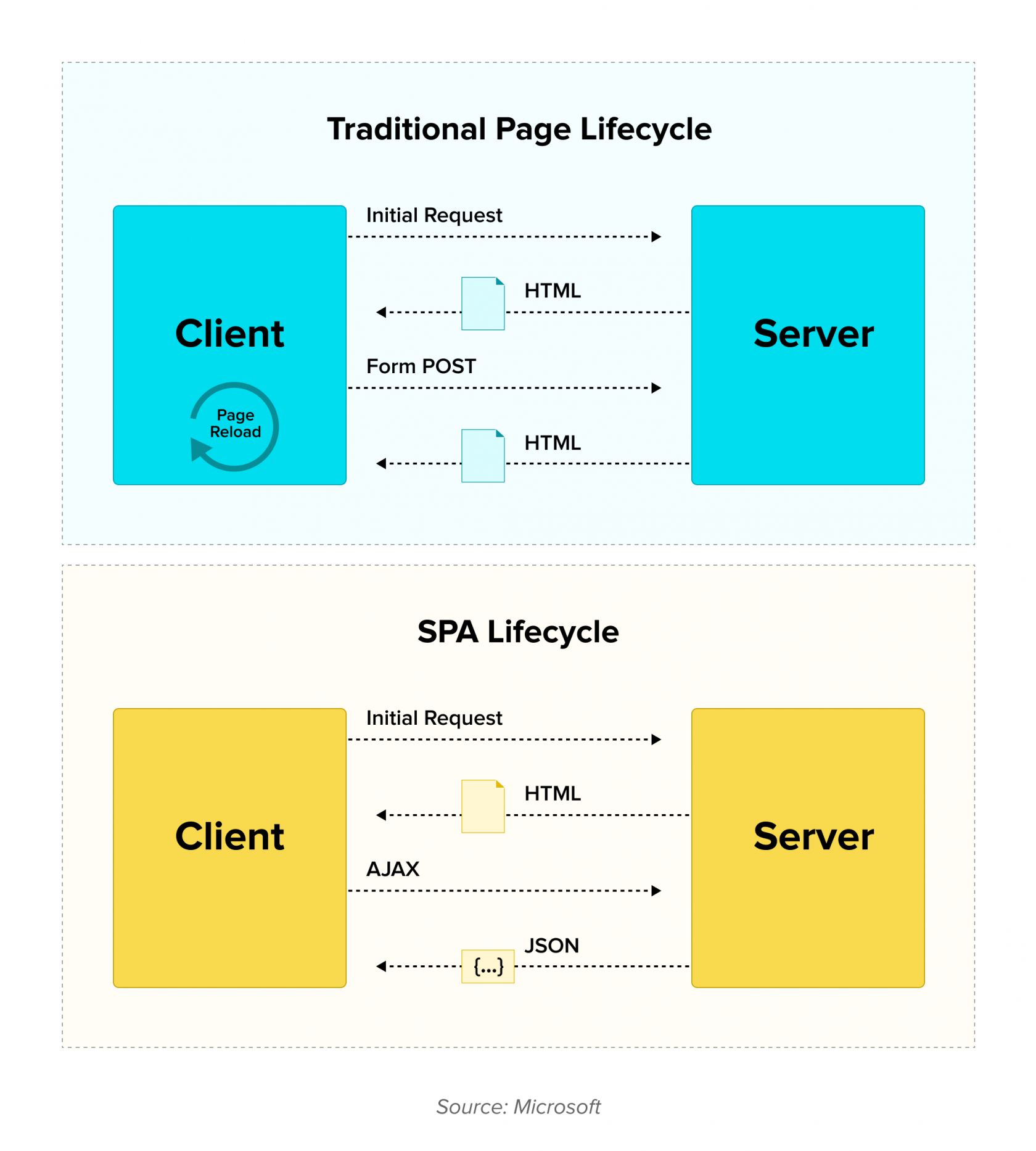


Рисунок 1.4.3 – *Життевий цикл MPA*

MPA підходить для великих компаній, що пропонують широкий спектр послуг і товарів: інтернет-магазини, сайти компаній, каталоги і торгові платформи.

В цілому, MPA є головною сторінкою і багато посилань на інші сторінки. Цей принцип використовується сайтами електронної комерції, такими як Ebay, Ozon і Amazon, де необхідно відображати велику кількість контенту.

Переваги MPA

* Проста SEO-оптимізації. MPA часто використовують сайти, для яких важливо потрапити в топи пошукових систем. Кожна зі сторінок має унікальну URL-адресу і є стабільною, що дозволяє пошуковим ботам адекватно її повзати.
* Масштабованість. Ви можете додати стільки інформації, скільки вам потрібно, до програми MPA, без обмежень на сторінку або функції.
* Перевірена класика. МПА працюють за тими ж принципами, що і веб-сайти з класичною навігацією, які знайомі користувачеві.

Недоліки МПА

* Повільна швидкість розвитку. MPA-додатки вимагають використання фреймворків з обох сторін: клієнта і сервера. Також через тісний зв'язок між інтерфейсом і бекендом робота розробників ускладнюється. Все це разом збільшує бюджет і час розвитку.
* Вартість створення мобільного додатку. Програми MPA не перетворюються добре на мобільні пристрої. Для цього в значній частині випадків буде потрібно розробка бекенда з нуля.
  + 1. **Що таке PWA-застосунок**

Progressive Web Application (PWA) це щось середнє між веб-сайтом і нативним додатком. Щоб почати роботу з PWA, користувач повинен завантажити та інсталювати сайт як застосунок. Подальший доступ до сайту здійснюється через іконку на робочому столі користувача. При натисканні сайт швидко відкривається без посередника у вигляді браузера.

PWA додатки цікаві тим, що вони:

* кросплатформні - робота відразу на декількох операційних системах;
* встановлюються в один клік, не виходячи на ринки додатків;
* завжди під рукою у користувача — сайт доступний за іконкою в смартфоні навіть в автономному режимі;
* зниження конкуренції в пошукових системах;
* дозволяють надсилати push-сповіщення як застосунок.
* легко конвертувати з існуючого сайту, просто змінити його трохи.

Прогресивні веб-додатки корисні для часто поновлювальних сервісів, до яких клієнти будуть продовжувати повертатися. Push-повідомлення дозволять періодично привертати увагу користувачів і заохочувати їх відвідувати застосунок, збільшуючи прибуток.

* 1. Використання фреймворків

JS-framework’и — є інструментами для побудови динамічних веб / мобільних / настільних додатків на мові Javascript. Як і з будь-якими іншими інструментами, розробники вдаються до використання js-фреймворків там, де неможливо / дуже складно / дуже довго виконувати завдання звичайними засобами. У переважній більшості випадків фреймворки використовуються для написання так званих односторінкових додатків.

З їх допомогою можна розробити як повноцінні сайти, так і функціональні модулі (різні онлайн-інструменти). Звичайно, для першого завдання краще підходять повноцінні фреймворки, а для другого рекомендується використовувати більш легкі фреймворки або бібліотеки.

Framework’и забезпечують чітку структуру програми і реалізуються за допомогою так званих «шаблонів проектування» (концепція більше мігрувала з бекенда, але явно має місце в зовнішній розробці з використанням js-фреймворків). Найбільш широко використовуються наступні шаблони: MVC (Model-View-Controller), MVP (Model-View-Presenter) та MVVM (Model-View-ViewModel).

Переваги побудови додатку на фреймворку JS:

* може бути легко реалізований SPA (Single Page Application);
* використання фреймворку JS зобов'язує нас мати структуру програми (скажімо, сильне «ні» коду спагетті);
* код стає помітно менше і він чистіший, що позитивно позначається на швидкості розробки, а також на підтримку і усунення помилок в коді заявки;
* наявність структури має на увазі модульність програми, а це полегшує роботу над додатком кільком розробникам одночасно;
* Наступна перевага більше випливає з використання самого JavaScript, але значно посилюється при використанні фреймворку: можливість швидкого створення мобільного і / або настільного кросплатформного додатку з веб-версії за допомогою таких систем, як PhoneGap або Apache Cordova.

Зі істотних недоліків можна виділити тільки тимчасово неповну підтримку пошуковими системами, однак це завдання рідко збігається з завданням реалізації SPA (Single Page Application), тим більше, що провідні пошукові системи (як мінімум, Google), вже майже повністю вирішили цю проблему.

Багато додатків вже випущені на JS-фреймворки і цей сегмент тільки набирає обертів. Для прикладів виконаних проектів на Angular.js наприклад, виділяється цілий сайт builtwith.angularjs.org, списки сайтів і додатків з використанням інших фреймворків можна знайти внизу на головних сторінках офіційних сайтів: backbone.js, ember.js.

Програми- збирачі

Зазвичай проект передбачає модульну структуру, тобто файлів буде багато. Оскільки імпорт не працює в браузері, а «власний синтаксис» може бути використаний, збирачу доведеться конвертувати вихідні файли в зрозумілий і робочий для браузера код.

Тому збирач (зазвичай webpack) повинен мати свої налаштування, які новачкові важко зрозуміти. Тут допомагає Vue CLI, який працює з коробки. Але, якщо завдання полягає в тому, щоб зробити щось інше, то вам доведеться виконати всі налаштування самостійно. А для цього потрібно вивчити Webpack і бажано робити це до або паралельно з вивченням js-фреймворку.

По-перше, веб-компоненти

Модульний або компонентний підхід до розробки вже давно використовується практично скрізь, крім JS. Все, що є в JS зараз, насправді досить примітивні речі. Веб-компоненти - це технологія майбутнього, яка може змінити ситуацію і зробити веб-сайти повноцінними додатками.

Прикладом веб-компонента є тег VIDEO, який відображає досить складний візуальний компонент з елементами управління. Тобто ми вказуємо тільки тег і його параметри, а браузер перетворює його в щось більш складне.

Якщо не зайти в технічну складову веб-компонентів, то, по суті, вони надають розробнику можливість створювати окремі сутності (компоненти) за допомогою звичайних HTML і CSS (in scope!), і писати логіку програми в JavaScript.

Наприклад, нам потрібно розмістити слайдер jQuery на сторінці. Тепер потрібно підключити js-файл, css-стилі, які при цьому не повинні конфліктувати з іншими стилями сайту, а також написати код ініціалізації. З веб-компонентами це трохи простіше - це просто якийсь конкретний тег HTML (наприклад, MY-SLIDER) і набір необов'язкових атрибутів.

Але на даний момент підтримка веб-компонентів слабка, тому єдиний спосіб отримати щось подібне - заздалегідь побудувати проект в зручному для браузера коді. І тут ми підходимо до питання - наскільки js-фреймворк відповідає поняттю веб-компонентів.

Друга причина - реальна підтримка сайту

Якщо ви хочете побачити приклади сайтів, зроблених в React, Angular або Vue, то, швидше за все, це будуть якісь жалюгідні агрегати - просто пил в порівнянні з повноцінними сайтами в PHP.

Шум навколо фреймворків JS явно не корелює з їх фактичним використанням у веб-розробці. Отже, які сайти роблять на цих рамках?

Як правило, це якісь закриті розробки для конкретного клієнта (b2b), який готовий заплатити за таку роботу чимала грошей. Але справа не тільки в тому, що клієнти, як правило, погано розбираються в технологіях, але і в тому, що замовнику важливо не тільки створити проект, але і підтримати його далі.

Php сайт складається з декількох частин. Перший - це робота з базою даних, тобто отримання зразка у вигляді структурованих даних. Це найпростіша частина роботи. Потім дані переходять на шаблонний рівень, де вже відбувається формування HTML-виведення. Тут це не стільки php-програмування, скільки html-макет. Після цього настала черга CSS довести вихід до потрібного дизайну. Якщо вам потрібно змінити / додати елемент сторінки, цикл PHP-HTML-CSS потрібно буде повторювати знову і знову.

Підтримкою такого проекту стануть повноцінні фахівці, які роздадуться відразу в декількох технологіях. І це в якійсь мірі проблема персоналу, тому потрібно постаратися побудувати ділянку так, щоб структурні частини взаємодіяли один з одним тільки за допомогою якихось примітивних методів.

Цей підхід використовується в фреймворках JS. На рівні PHP надається якийсь API, запити на які повертають дані json. Ці дані отримані в js через ajax, а це означає, що js-програміст може взагалі не розуміти, як вони виникли - йому дали URL і приклад використання, а цього вже достатньо. PHP програмісту не потрібно вигадувати html-код і навіть думати про те, як будуть виглядати ці дані на сайті.

Сам макет відбувається на рівні компонентів - це вже HTML і CSS з примітивними включеннями js.

(React || Angular || Vue) vs. jQuery

Насправді, дуже дивне твердження питання, оскільки порівнювати ці фреймворки з jQuery дещо некоректно. Але якщо ви google, це питання є найбільш частим.

Потрібно розуміти, що jQuery - це просто набір функцій, які зібрані в одну оболонку (бібліотеку). Це не краще або гірше, ніж Vue або React, які є їх власними наборами функцій.

Питання тут скоріше в загальній ідеології. jQuery можна використовувати в будь-якому місці на сторінці і з будь-якими елементами. Це такий «атомний» підхід, який дозволяє використовувати будь-які елементи. JS-фреймворки припускають більш комплексний підхід і розглядають сторінку вже не як простий набір html-тегів, а як пов'язані компоненти. І ніхто не заважає використовувати jQuery в поєднанні з ними.

Існує ряд завдань, які легше вирішити за допомогою фреймворку, особливо якщо вони «заточені» для цих завдань. Але також набагато простіше використовувати готовий плагін jQuery, замість того, щоб винаходити колесо.

Типи сайтів для javascript фреймворків

JS-фреймворки, у зв'язку з тим, що вони працюють на стороні клієнта, не можуть використовуватися нормально там, де багато динамічних даних і різних сторінок / адрес, наприклад, в блогах. Тому основною метою для них є SPA (односторінковий застосунок), де вся дія відбувається на одній сторінці. І хоча для них вже є рішення у вигляді маршрутизації, це все одно виглядає дещо дивно.

Тому js-фреймворки будуть цікаві тільки там, де потрібен високий «динамізм», наприклад, в тій же gmail-пошті. Головна особливість тут полягає в тому, що інтерфейс змінюється без перезавантаження сторінки і виглядає як звичайна програма в цілому.

Js-фреймворк не повинен використовуватися на звичайних сайтах або цільових сторінках. Як правило, немає необхідності в декількох запитах Ajax, а значить, ви можете легко обійтися звичайним PHP / HTML / CSS / JS / jQuery.

1. АНАЛІЗ ТА ШЛЯХИ РОЗВ’ЯЗКУ ПРОБЛЕМИ  
   1. React – UI – Фреймворк
      1. Чому саме React

Приступаючи до роботи над новим веб-проектом, багато розробників задаються питанням: «Які інструменти найбільш підходять?».

Очевидно, [що ядром буде JavaScript,](https://merehead.com/ru/blog/best-javascript-conferences/) так як сьогодні він забезпечує найбільш широкий функціонал для створення інтерфейсу. Але дилема виникає при виборі середовища розробки, а саме [кращих javascript фреймворків,](https://merehead.com/ru/blog/ru-best-javascript-framework-and-libraries-to-learn-in-2021/) адже кожен наділений унікальними особливостями.

Як правило, суперечка будується навколо трьох рамок **–** **Angular.JS,** **React.js і** **Vue.js**. Тому зараз ми розглянемо їх переваги і недоліки, а також популярність серед розробників.

Вибір розробників

Stackoverflow

Перш за все, ми проаналізували тематичні опитування, [в тому числі State JavaScript](https://2019.stateofjs.com/front-end-frameworks/) і [Stackoverflow.](https://insights.stackoverflow.com/survey/2019" \l "most-loved-dreaded-and-wanted" \t "_blank) Щороку вони надають найбільш точну інформацію про популярність і статус розвитку різних середовищ розвитку.

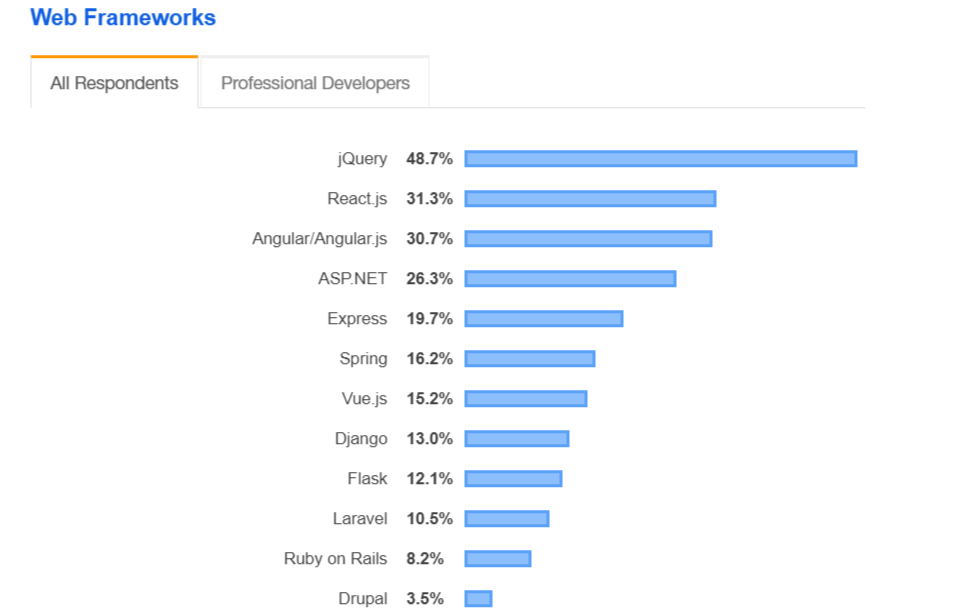


Рисунок 2.1.1 – *Найпопулярніші веб-фреймворки*

Незважаючи на стрімкий розвиток Vue.js, ця структура займає лише 7-е місце в рейтингу (як серед всіх респондентів, так і серед професіоналів). **React.js** і **Angular.js**розділили 2-е і 3-е місця.

Примітно, що за загальним результатом React.js займає 2-е місце, але тільки професійні розробники вивели Angular.js на 2-е місце.

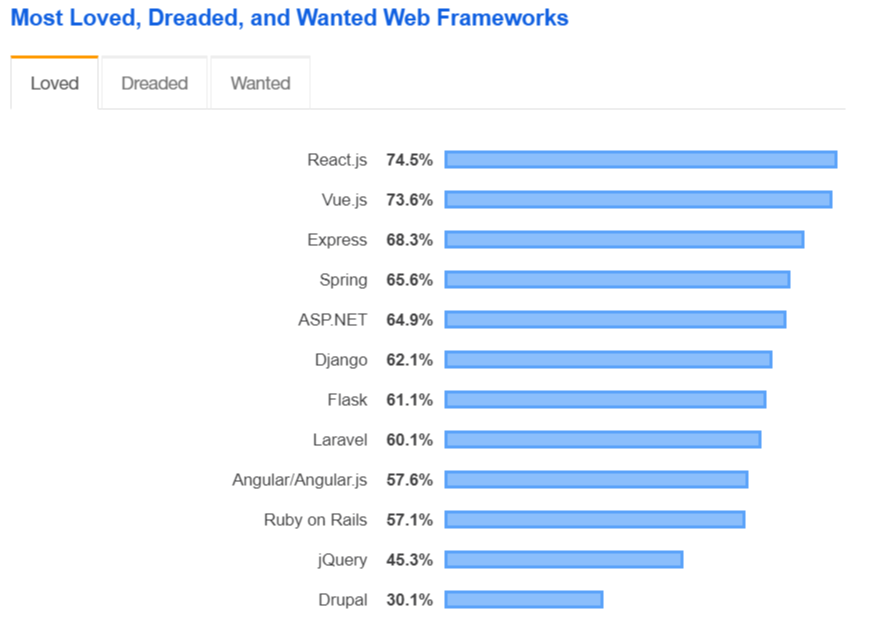


Рисунок 2.1.2 – *Найбільш переважні фреймворки*.

У той же час, розробники люблять найбільше працювати з **React.js і Vue.js, коли Angular.js**був майже в кінці списку. Серед тих, хто раніше не розробляв додатків з використанням цих фреймворків, найбільш бажаним для вивчення був React.js. За нею йде Vue.js, а третє місце займає **Angular**.js.

NPMtrends

Одним з важливих показників розвитку технологічного стека або середовища розробки є кількість завантажень пакетів. **Тенденції NPM** показують найбільш точну статистику для кожного інструменту, включаючи не тільки кількість завантажень, але і дані Github.

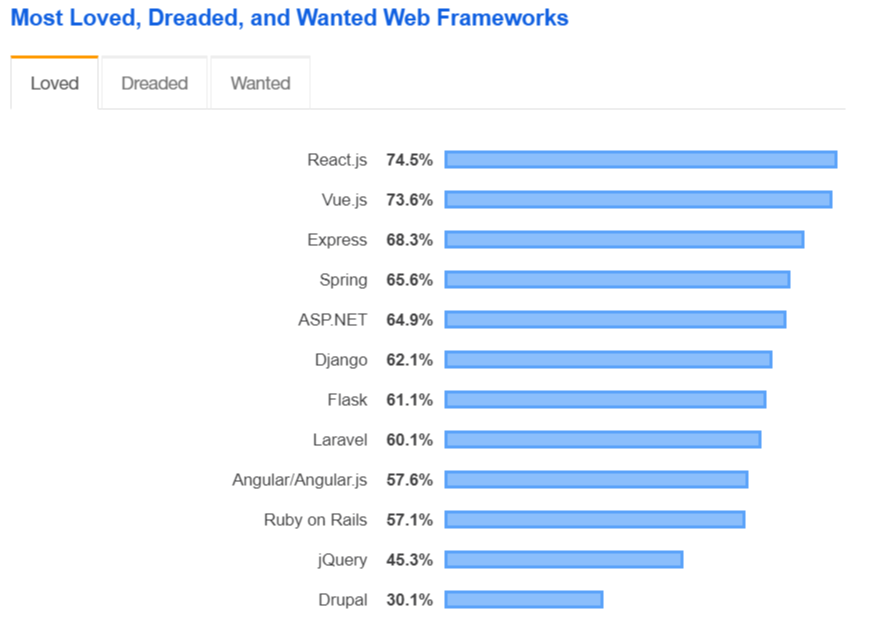


Рисунок 2.1.3 – *Кількість завантажень Angular.js, React.js та Vue.js*

Як бачите, React.js значно випереджає своїх конкурентів у цьому плані. Angular.js також втратив значну кількість завантажень у порівнянні з 2018. З середини 2019 року вона не перевищила позначку в 1 мільйон. **Vue.js** поступово набирає обертів. Так, на початок 2019 року у нього було не більше 500 тисяч завантажень, тоді як сьогодні показник стабільно перевищує 1 мільйон.

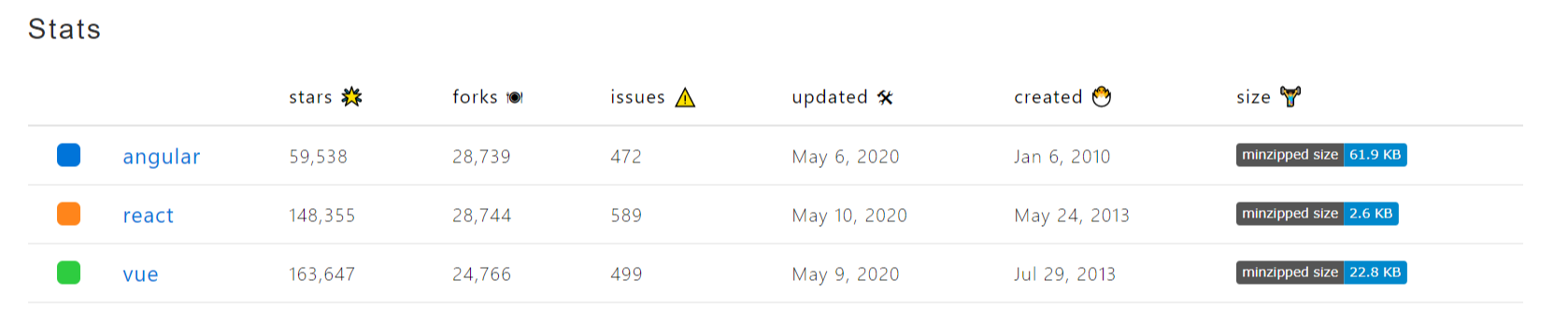


Рисунок 2.1.4 – *Рейтинг фреймворків на Github*

Ці цифри узгоджуються з результатами опитування Stackoverflow. Рейтинг Vue.js і React.js значно вище, ніж Angular.js. Це і є причиною великої різниці в кількості завантажень пакетів.

State JavaScript

Даний річний звіт є основним показником розвитку **JavaScript** в цілому і всіх інструментів, які з ним пов'язані.

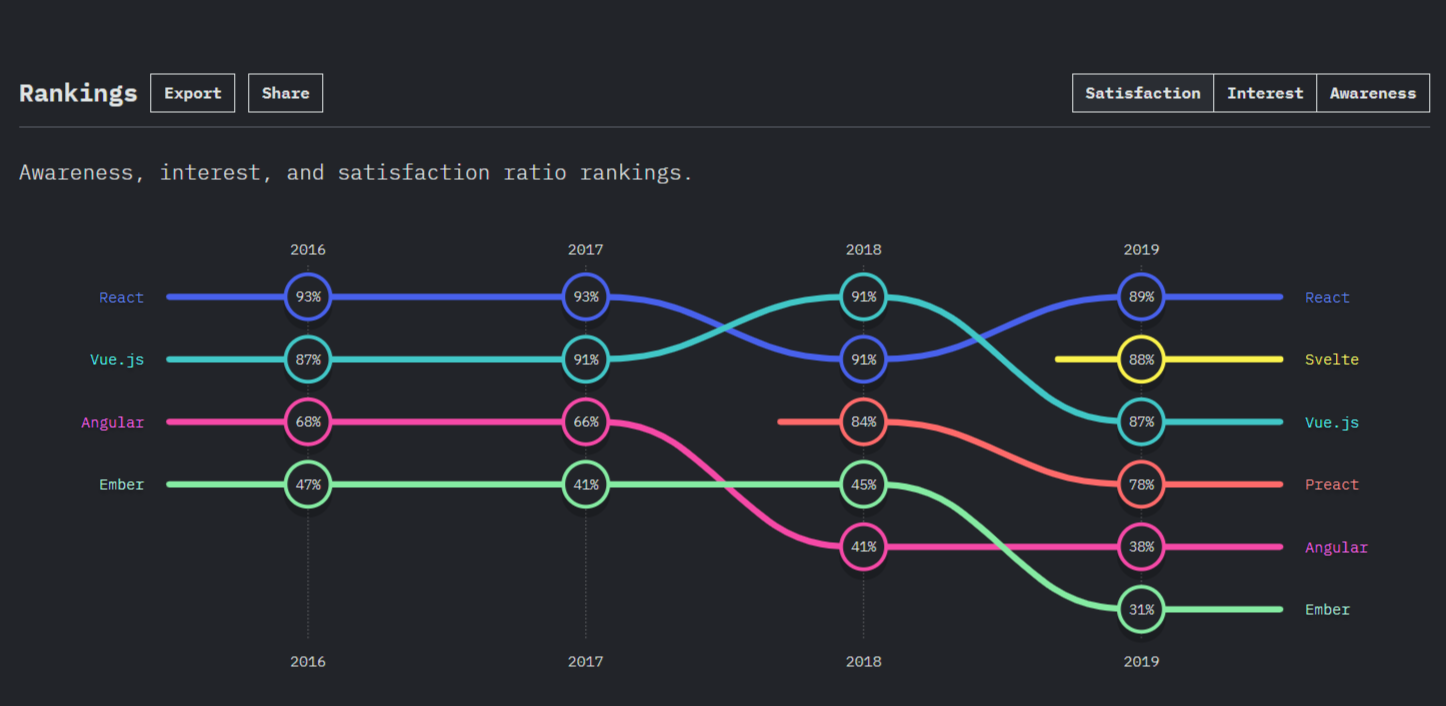


Рисунок 2.1.5 – *Рейтинг фреймворків за станом JavaScript 2019*

В основному розробники задоволені робочим процесом з React.js і Vue.js, в той час як Angular.js задовольняє їх лише на 38%. Точно так само лише 23% розробників мають особливий інтерес до Angular.js, тоді як Vue.js і React.js цей показник становить 64% і 61% відповідно.

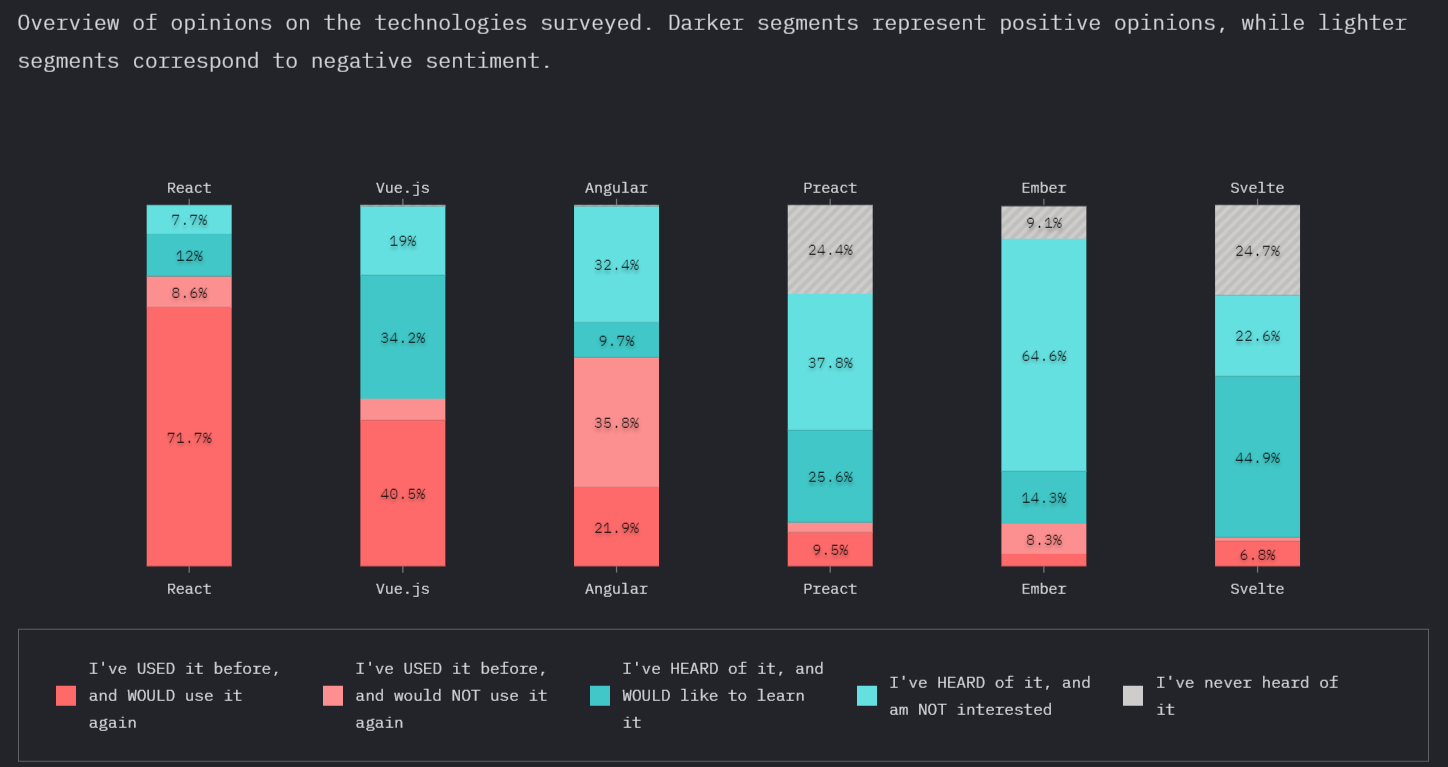


Рисунок 2.1.6 – *Ставлення до фреймворків*

Що стосується рамок, то аналогічні результати. Більшість респондентів або не цікавляться **Angular.js,** або раніше використовували його, але більше не хочуть з ним працювати. React.js також дуже популярний, більшість респондентів продовжують використовувати цю рамку для вирішення проблем.



Рисунок 2.1.7 – *Статистика по використанню фреймворків в залежності від розміру компанії.*

Примітно, що відносно невеликі компанії віддають перевагу **React.js** і **Vue.js,** в той час як великі розробники (від 100 і більше 1000 співробітників) частіше використовують Angular.js. Однак ці цифри приблизно рівні.

Переваги, недоліки, особливості

Тепер давайте проаналізуємо кожну рамку окремо:

* У яких випадках вибір буде неочевидним?
* Чому розробники вибирають React.js?
* З якими труднощами ви зіткнетеся з кожним середовищем розвитку?

Навіщо **React**.js?

**React.js** третій рік поспіль займає перший рядок з усіх рейтингів. Деякі розробники не мають претензій до цієї структури, так як вона стрімко розвивається і стає більш стабільною.

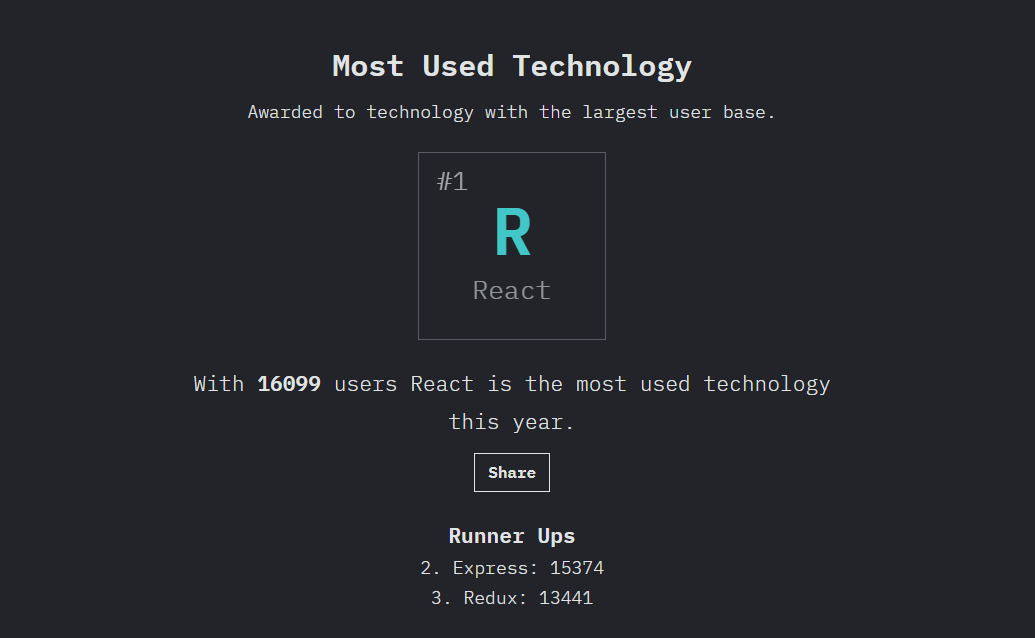
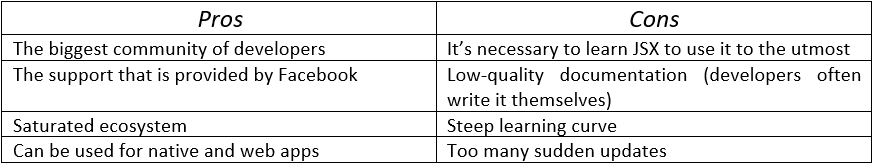


Рисунок 2.1.8 – *React.js є найбільш часто використовуваним інструментом JavaScript.*

Примітно, що насправді **React.js** - це не фреймворк, а бібліотека. Але він має такий широкий функціонал, що часто інструмент можна використовувати без додаткових інтеграцій. Передня кінець в результаті буде не менш ефектним.

**React.js** створена командою Facebook в 2013 році з метою поділу користувацького інтерфейсу на набір компонентів для спрощення процесу розробки.

Таблиця 2.1.1 – Переваги, недоліки **React**.js



**React.js** є дуже багатим середовищем розвитку. Здебільшого користувачі реагують на нього тільки з позитивного боку.

Останнім часом, однак, **Facebook** випускає занадто багато оновлень, тому деякі інструменти стають застарілими. Деяким розробникам подобається такий хід подій. Але здебільшого суспільство скаржиться на це. Адже таким чином їм потрібно постійно опановувати нові методи.

Примітно, що **сьогодні фахівці React. JS є найбільш затребуваними в США.** Станом на початок травня 2020 року на LinkedIn на вимогу React.js є понад 30 000 відкритих позицій, а на Indeed – близько 4 000.

Однією з ключових причин популярності React.js є **підтримка авторитетної компанії Facebook.** На основі цього середовища розробки сьогодні працюють такі сервіси, як Instagram, WhatsApp і Twitter. Всі вони дуже швидкі, привабливі за зовнішнім виглядом. Це забезпечило високий рівень довіри до React.js.

Незважаючи на безліч переваг, є лише кілька випадків, коли це середовище розробки буде найбільш підходящим:

* швидка розробка легких додатків корпоративного рівня
* створення SPA або кросплатформних додатків;
* розширення функціональності існуючого додатку.

Це не означає, що React**.js** погано в іншому випадку. Швидше за все, альтернативи були б кращими.

Чому Vue.js?

Vue.js стало відкриттям останніх років. Несподівано зі звичайного середовища розвитку він став одним з найулюбленіших серед професіоналів.

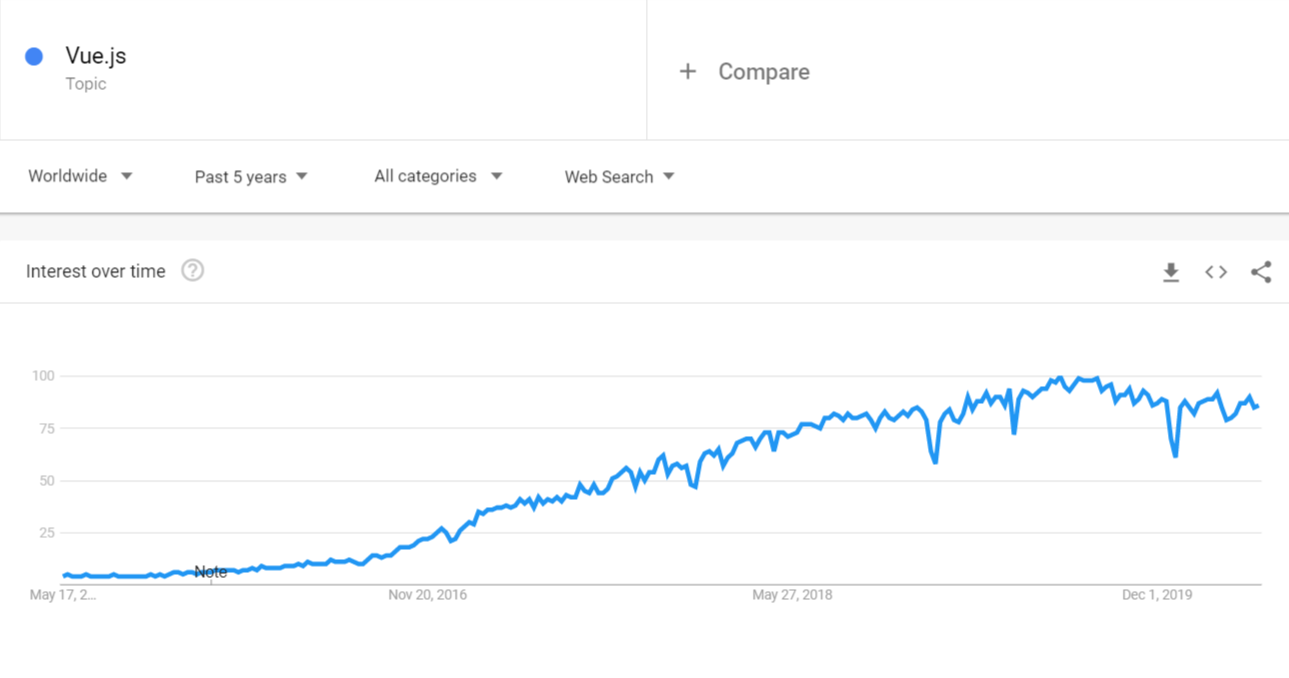
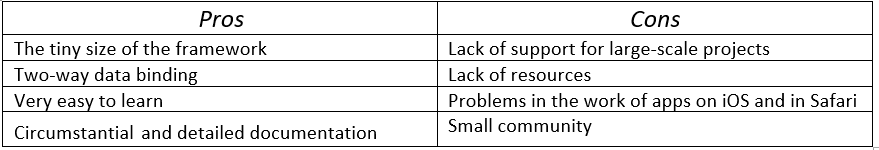


Рисунок 2.1.9 – *Динаміка популярності Vue.js*

Примітно, що популярність зросла без підтримки великих компаній. Гіганти ринку звернули увагу на Vue.js тільки після різкого стрибка. Саме тому вона виявилася третім фреймворком в тривалому **протистоянні React.js і Angular.js.**

Таблиця 2.1.2 – Переваги, недоліки Vue.js



Vue.js зараз виділяється тим, що має величезну кількість особливостей. Деякі з них можуть навіть **виступати в якості недоліків.** Наприклад, Vue.js дуже гнучкий. Але в разі роботи над великими проектами у великій команді це може сприяти більшій кількості помилок.

Після того як **Vue** почав показувати свої характеристики, на нього звернули увагу багато гігантів ринку, в тому числі Gitlab, WizzAir, EuroNews. Grammarly повністю побудована на вершині Vue, а Alibaba і Xiaomi в 2018 році оголосили про повний перехід на Vue.js.

Крім того, сьогодні **фреймворки** користуються великим попитом на азіатському ринку. Тому переважна кількість обговорень і хитрощів використання описана китайською мовою. Не знаєте мови? Вам доведеться обмежитися документацією та рекомендаціями англійською мовою.

Незважаючи на те, що середовище розробки раптом стало популярним, попит на розробників не збільшився в тій же мірі. Так, на **LinkedIn** є всього 8000 відкритих вакансій на замовлення Vue.js, а на Indeed - близько 4000.

У яких випадках Vue.js буде кращим вибором?

* розробляти розумні та високопродуктивні додатки;
* щоб отримати застосунок на ринок раніше.
* створюйте невеликі та легкі програми, такі як Grammarly.

Чому **Angular**.js?

**Angular.js** вже давно є найкращим вибором для розробки інтерфейсу користувача. Таким чином, він став частиною популярного стека MEAN.

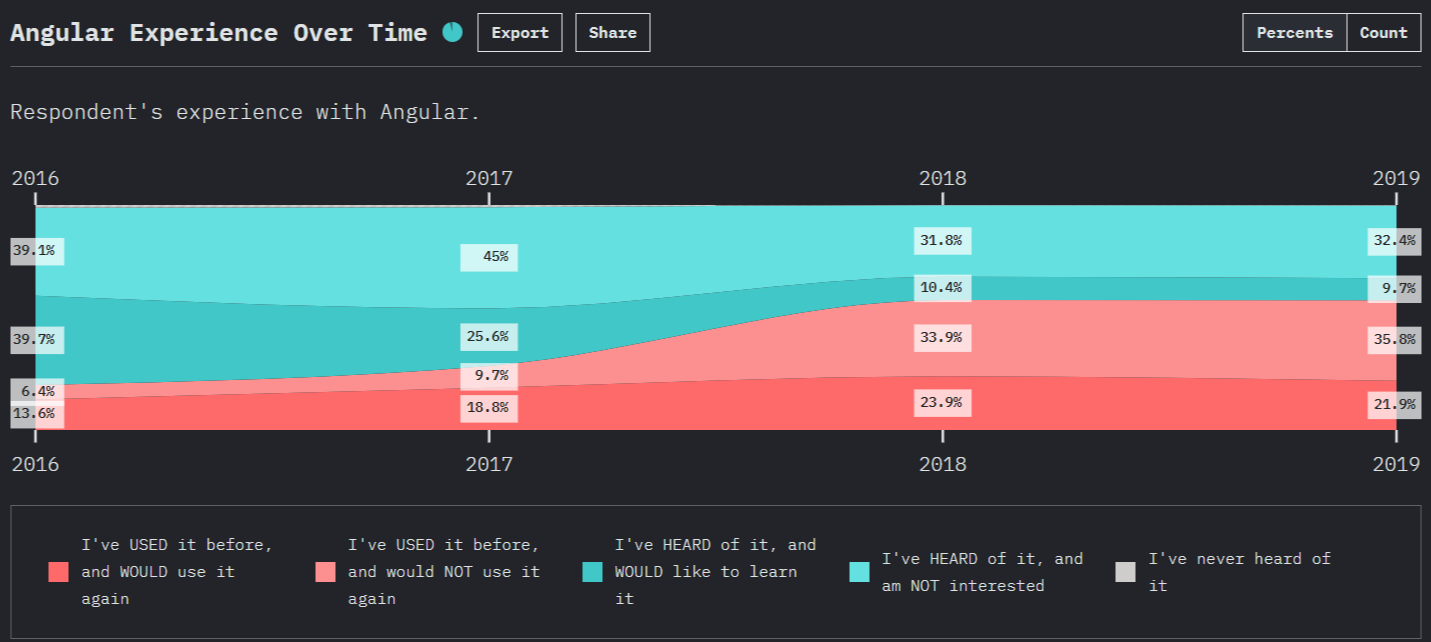
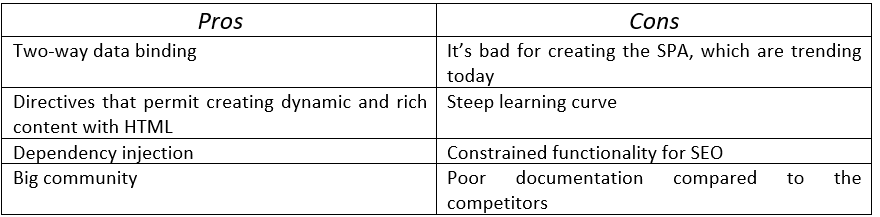


Рисунок 2.1.10 – Динаміка розвитку інтересу до Angular.js

Тим не менш, за останні кілька років розробники все частіше скаржаться на неповноцінність фреймворку в порівнянні з конкурентами. В результаті багато компаній поступово відмовляються від використання **Angular.js.**

Таблиця 2.1.3 – Переваги, недоліки **Angular**.js



Багато в чому фреймворк втратив свою популярність через появу нових тенденцій веб-розробки. Команда Angular.js не реалізувала необхідний функціонал в нових версіях середовища розробки. Тому сьогодні ми бачимо, що Vue.js і React.js стають більш кращим вибором. Виходячи за рамки цих трьох рамок, кутовий іноді поступається іншим інструментам (наприклад, Svelte).

Тим **не менш**, Angular.js як і раніше використовується для підтримки багатьох популярних веб-сайтів і веб-додатків. Серед них були The Guardian, UpWork, PayPal і Sony. Всі вони є великими ділянками, в яких Angular.js проявляється досить добре.

Найцікавішим фактом є попит на Angular.js фахівців. Незважаючи на зниження популярності і погіршення рейтингів, попит практично нарівні з **React.js.** Так, на LinkedIn налічується близько 30 000 відкритих вакансій, а на Indeed - понад 5 000.

У яких випадках слід звернути увагу на кутовий.js?

* створення великомасштабних додатків;
* необхідність легко масштабованої архітектури;
* створення месенджерів та інших додатків «в режимі реального часу»;
* написання коду з

Висновок

Грунтуючись на статистиці, результатах опитувань і різних звітах, можна зробити висновок, що найбільш бажаним вибором в 2021 році є React.js. Друге місце займає **Vue.js,** а **Angular.js** залишиться на третьому з великим відставанням.

Але якщо абстрагуватися від статистики і враховувати тільки ситуативне застосування, то **кращих фреймворків** на сьогоднішній день точно немає. Так чи інакше, кожне середовище розробки має свої переваги і недоліки. Точно так само кожен з них має чималу кількість шанувальників і противників.

* + 1. React Virtual DOM

DOM розшифровується як Document Object Model. Маніпуляція DOM дуже важлива для сучасних інтерактивних веб-технологій. Його часто називають серцем сучасної мережі. Це є абстракцією структурованого тексту. Але він працює повільніше, ніж інші операції JavaScript, оскільки більшість фреймворків JavaScript зазвичай оновлюють DOM, вони це роблять навіть якщо не потрібно це робити. Це означає, що ці оновлення не обов'язково потрібні для виконання дії, але вони все ще виконуються за замовчуванням. Наприклад, припустимо, що дев'ять товарів було покладено в кошик в Інтернет-магазині. Тепер скажемо лише перший необхідний для покупки та переходу до оплати. Тут більшість технологій відновлює весь список, який був поміщений у кошик. Це означає, що система повинна лише через одну зміну перебудувати всю сторінку, як саме було раніше.

React не винайшов віртуальний DOM, а використовує та надає його спільноті розробників безкоштовно. Virtual Dom – це просто абстракція HTML DOM. React має відповідний віртуальний DOM-об'єкт для кожного DOM-об'єкта. Віртуальний DOM також характеризується властивостями, подібними до справжнього DOM. Однак він не може вносити зміни безпосередньо до HTML. Маніпуляція з DOM є досить повільним процесом. Але маніпулювання віртуальним DOM відбувається швидше, оскільки це не має нічого спільного з частиною відображення і не вносить жодних змін на екран. Рисунок 1, є ілюстрацією віртуального DOM в пам'яті.

Як показано на рисунку 1, React virtual DOM в пам'яті є полегшеною копією справжнього DOM. React використовує метод, при кожній зміні віртуального DOM оновлюється потрібний JSX елемент. Це може здатися неефективним, але насправді віртуальний DOM досить швидко оновлюється і не робить ніякого впливу в процес. Після запиту на оновлення DOM, React порівнює старий стан і новий та визначає який елемент було змінено. Як тільки React виявляє змінені елементи, він оновлює лише ці об'єкти до реального DOM.

Таким чином, React пришвидшує оновлення сторінки за допомогою Virtual DOM. У вищезазначеному прикладі, React оновлював би лише позначений елемент зі списку і залишав решту елементів недоторканими. Цим і відрізняється оновлення сторінки в додатку, в той час коли React змінює тільки необхідні частини в DOM. Цей процес маніпулювання віртуальним DOM є однією з головних причин, чому React набирає великої популярності серед спільноти розробників

Серед багатьох переваг бібліотеки ReactJS є декілька ключових переваг:

* Алгоритми в React досить швидкі та ефективні
* Включення JSX та гіперскрипту дозволить нам створити кілька інтерфейсів для одного і того ж застосунку.
* Він дуже легкий і може працювати на будь-якому мобільному пристрої.
* Це обізнаність споживачів про продукт та багато способів застосування.
* Його також можна використовувати без React як самостійний двигун
* Деякі недоліки React полягають у наступному:
* Він займає значну частину пам'яті. Повна в пам'яті копія DOM.
* Статичні та динамічні елементи не приносять особливої різниці.
  + 1. React router dom

У будь-якому реальному веб-додатку потрібні маршрути, і застосунок React не є винятком. Користувач повинен побачити, де вони знаходяться в додатку в будь-який момент часу. І він бачить своє поточне місцезнаходження в адресному рядку браузера. Таким чином, ваш застосунок повинен мати можливість зіставити певну URL-адресу з відповідною сторінкою. Тобто, якщо ми вводимо в адресний рядок, наприклад , то застосунок повинен направити нас на сторінку списку прийомів, але не на будь-який інший.

Історія також повинна працювати. Тобто, коли користувач натискає на стрілку «Назад» в браузері, застосунок повинен направити нас на попередню сторінку.

React сама по собі не надає такої можливості, це завдання спеціальних бібліотек. Як правило, використовуючи API такої бібліотеки, підключаємо компоненти сторінок нашої позиції, зіставляючи їх з певними шляхами. Після цього, переходячи з однієї сторінки на іншу, ми побачимо в адресному рядку, як змінюється поточний шлях.

На даний момент існує кілька популярних бібліотек для маршрутизації. Найпопулярнішою є React-router-dom. Ця бібліотека досить проста у використанні і має [хорошу документацію.](https://reacttraining.com/react-router/web/guides/quick-start)Вона надає такі функції, як:

* Навігація клацанням миші (компонент <Link>)
* Переспрямування (компонент <Redirect>)
* Маршрутизація (компонент <Route>)
* Історія (властивість history)

Це лише мала частина того, що може зробити бібліотека.

* 1. Redux – Менеджер станів додатку
     1. FLUX-архітектура

Facebook розробив Flux-архітектуру як спробу вирішити проблеми, викликані шаблоном MVC (Model-View-Controller) у своїй масивній кодовій базі. Вони боролися з проблемами, коли дії призвели до каскадних оновлень, що призвело до непередбачуваних результатів і коду, який стало важко відлагоджувати. Це може здатися знайомим, якщо ви використовували MVC фреймворки раніше, оскільки більшість з них по суті дуже схожі. При додаванні спостерігачів і двосторонніх даних, що зв'язуються з сервером, і ви отримаєте справжній головний біль.

Flux-архітектура – архітектурний підхід або набір патернів проєктування для побудови користувацького інтерфейсу веб-додатків, в поєднанні з реактивним програмуванням і побудований на односпрямованих потоках даних.

Згідно з планом творців, і незважаючи на те, що Facebook забезпечив реалізацію Flux на застосунок до React, Flux - це не ще одна веб-база, а архітектурне рішення.

Головною відмінною рисою Flux є односторонній напрямок передачі даних між компонентами архітектури Flux. Архітектура накладає обмеження на потік даних, зокрема, виключаючи можливість оновлення стану компонентів самостійно. Такий підхід робить потік даних передбачуваним і полегшує відстеження причин можливих помилок в програмному забезпеченні.

Порівняння Flux та MVC архітектур.

Щоб краще описати потік, ми порівняємо його з однією з провідних клієнтських архітектур: MVC. У клієнтському застосунку MVC  взаємодія користувача запускає код у контролері. Контролер знає, як координувати зміни в одній або декількох моделях, викликаючи методи на моделях. Коли моделі змінюються, вони повідомляють одне або кілька переглядів, які, в свою чергу, читають нові дані з моделей і оновлюють себе відповідно, щоб користувач міг бачити ці нові дані.

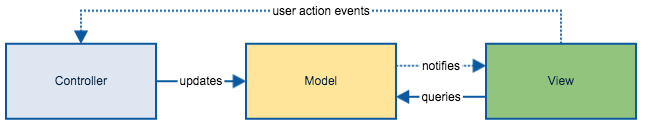


Рисунок 2.2.1 – *Схематичне зображення простого додатку MVC-архітектури*

Оскільки програма MVC зростає, а контролери, моделі та подання додаються, залежності стають складнішими.

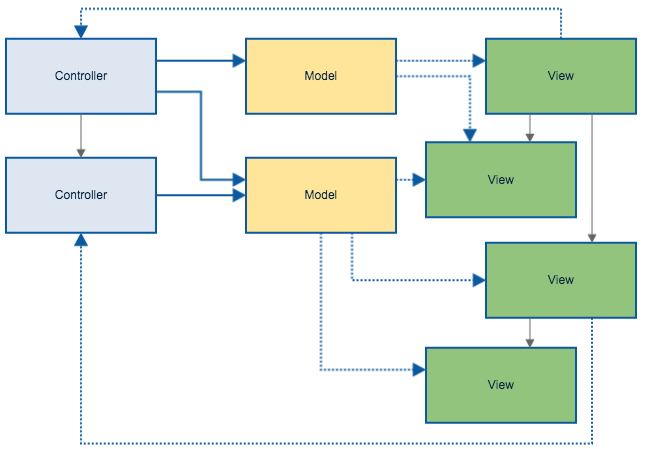


Рисунок 2.2.2 – *Схематичне зображення складного додатку MVC-архітектури*

З додаванням всього трьох переглядів, одного контролера і однієї моделі, графік залежності вже важче простежити. Коли користувач взаємодіє з інтерфейсом користувача, виконуються кілька шляхів коду розгалуження, і проблеми з налагодженням у стані програми стають вправою для з'ясування того, який модуль (або модулі) в одному (або декількох) з цих потенційних шляхів коду містить помилку. У гіршому випадку взаємодія з користувачем спричинить оновлення, які, в свою чергу, спричиняють додаткові оновлення, що призводить до помилок та каскадних ефектів, схильних до помилок, уздовж кількох з цих, іноді перекриваються, шляхів.

Flux уникає цієї конструкції на користь односторонній потік даних. Усі взаємодії користувачів у поданні викликають *дію (Action Creator)*, що призводить до того, що подія *Action*  викликається диспетчером (*Dispatcher*). Диспетчер є однією точкою виклику для всіх дій у Flux-додатку. Дія відправляється від диспетчера в  *сховище (Store)*, які оновлюються у відповідь на дію.

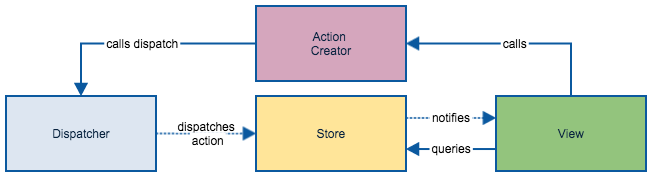


Рисунок 2.2.3 – *Схематичне зображення простого додатку Flux-архітектури*

Потік істотно не змінюється для додаткових станів або виглядів (*View*). Диспетчер просто відправляє всі дії в *усі* стани в додатку. Зверніть увагу, що він не містить знань про те, як насправді оновити стан – саме сховище містить цю бізнес-логіку. Кожен стан відповідає за домен програми, і оновлює себе тільки у відповідь на дії.

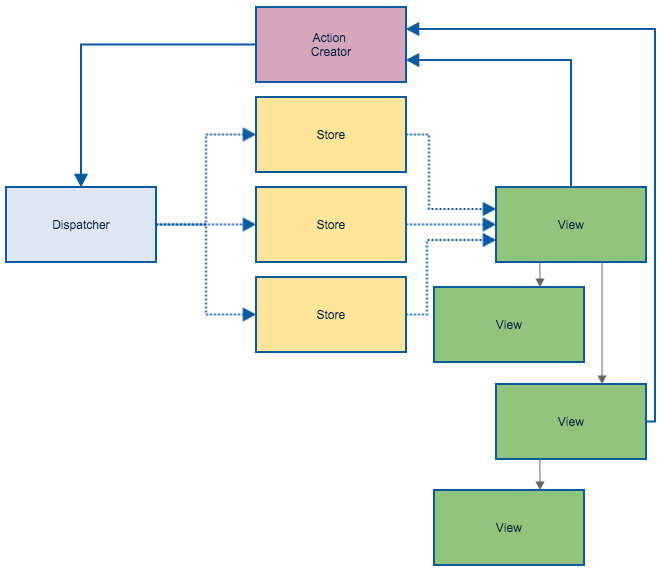


Рисунок 2.2.4 – *Схематичне зображення складного додатку Flux-архітектури*

Коли стан оновлюється, він випускає подію зміни. У багатьох додатках React спеціальні перегляди (відомі іноді як "контролер-перегляди") відповідають за спостереження за цією подією змін, читання нових даних станів та передачу цих даних через властивості до дочірніх переглядів. Це не рідкість у програмі React для події зміни стану, щоб викликати повторний рендеринг подання верхнього рівня, ефективно ре-рендерить всю ієрархію подання (яка ефективно обробляє React). Це повністю дозволяє уникнути складних помилок і проблем з продуктивністю, які можуть виникнути через спроби спостерігати за конкретними змінами властивостей моделей і лише незначно змінювати частини переглядів.

Архітектура потоку має кілька ключових властивостей, які роблять її унікальною і забезпечують важливі гарантії, всі з яких зосереджені навколо того, щоб зробити потік даних явним і легко зрозумілим і збільшити локалізацію помилок.

* Примусова синхронність. Дії (*Actions*) та їх обробники (*Handlers*) всередині сховища синхронні. Усі асинхронні операції мають викликати дію, яка повідомляє системі про результат операці. Хоча дії можуть робити асинхронні запити на сервер (API-call), обробники дій у сховищі в ідеалі цього не зроблять. Це правило робить потік інформації в додатку надзвичайно явним; налагодження помилок у стані застосунку просто включає в себе з'ясування того, яка дія викликала поганий стан, а потім знайти неправильну логіку, яка відповіла на цю дію.
* Інверсія контролю. Оскільки стани оновлюються всередині сховища у відповідь на дії (а не оновлюються ззовні контролером або подібним модулем), жодна інша частина системи не повинна знати, як змінити стан програми. Вся логіка оновлення стану міститься в самому сховищі. І, оскільки сховище тільки коли-небудь оновлюються у відповідь на дії, і тільки синхронно, тестування сховища просто стає питанням: поставити їх у початковий стан (INITIAL\_STATE), надіслати їм дію та перевірити, чи змінився правильно стан.
* Семантичні дії. Оскільки сховище повинно оновлювати себе у відповідь на дії, дії, як правило, семантично описові. Наприклад, у Flux-програмі форуму, щоб позначити повідомлення як прочитане, можна надіслати дію з відповідним типом . Дія (і компонент, що генерує дію) не знає, *як* виконати оновлення, але *описує,* що він хоче, щоб сталося. MARK\_THREAD\_READ. Через цю властивість рідко доводиться змінювати типи дій, лише те, як на них реагує сховище. До тих пір, поки ваша програма має концепцію "потоку", і у вас є кнопка або інша взаємодія, яка повинна позначати повідомлення як прочитане, тип дії семантично дійсний. MARK\_THREAD\_READ
* Без каскадних дій. Потік забороняє відправку другої дії в результаті відправки дії. Це допомагає запобігти каскадним оновленням, які важко налагоджують, і допомагає вам думати про взаємодію у вашому додатку з точки зору семантичних дій.
  + 1. Redux

‎ Redux був розроблений Деном Абрамовим і Ендрю Кларком. Спочатку він був випущений в 2015 році. Він написаний на JavaScript. В основному використовується з бібліотеками для створення користувацьких інтерфейсів для фронтових технологій, таких як React, Angular і т.д. Він підтримує кросплатформленість.‎

‎Redux також називають інтуїтивно зрозумілим державним контейнером. Він також може бути використаний з React і іншим видом бібліотеки. Redux дуже простий, якщо ви знаєте Flux або ви розробили щось на Flux.‎

‎Redux в основному використовується там, де розумна кількість даних змінюється з плином часу. Якщо дані не часто змінюються, то використання Redux менше. Він використовується як шаблон для управління станом програми.‎

‎Деякі бібліотеки або фронтові технології, такі як React, мають власне управління станом додатків, тому вам слід вивчити їх вбудовані можливості при використанні цих бібліотек. Іноді після розробки додатку стає важко зрозуміти і закодувати, важко зрозуміти, як змінилася держава. У цьому випадку Redux є корисним і використовується.‎

* + 1. Thunk мидлвар

‎

За замовчуванням дії Redux обробляються синхронно, що є проблемою для будь-яких користувацьких додатків, яким потрібно взаємодіяти із зовнішніми API або використовувати побічні ефекти. Redux також дозволяє використовувати проміжне програмне забезпечення між обробкою дії та дією, яка досягає коробок передач.‎

‎Є дві дуже популярні бібліотеки середнього програмного забезпечення, які підтримують побічні ефекти та асинхронні дії: ‎‎Redux Thunk‎‎, ‎‎Redux Saga та Redux Observable.‎‎

‎Thunk,‎‎ або converter, - це концепція програмування, в якій функція використовується для затримки оцінки або розрахунку операції.‎

‎Redux Thunk - це проміжне програмне забезпечення, яке дозволяє викликати творців дій, які повертають функцію замість об'єкта дії. Ця функція отримує метод обробки сховища, який потім використовується для обробки регулярних синхронних дій в тілі функції після виконання асинхронних операцій.‎

* 1. Статична типізація пропсів у React додатку

Компоненти React – по суті це функції JavaScript, які повертають елементи React, а саме JSX. Вони визначають, що буде додано до DOM. Так само, як і функції, компоненти React можуть отримувати аргументи, які називаються пропсами, що може призвести до динамічних повернених елементів.

Пропси можуть бути будь-якого типу даних, але очікуваний тип даних для компонентів може відрізнятися. Коли проект дуже великий розробник може вже забувати деталі свого давнього коду, або зовсім – чужого. Наприклад компонент «Таймер» приймає пропсу «Кінцева дата». Вона може бути   
строкою ("24.08.22 12:30"), датою ("1995-12-17T01:24:00.000Z") або взагалі  
числом (23049818239). Щоб не помилитися треба або детально вивчити компонент, або чекати помилку під час виконання програми.

Щоб цього запобігти та бачити у підказках IDE чого чекає від нас той чи інший компонент є низка бібліотек та навіть мов програмування. Розглянемо два способи, як це зробити PropTypes і TypeScript. Незважаючи на схожі цілі, ці два методи відрізняються тим, як вони працюють.

Що таке PropTypes?

PropTypes - це інструмент перевірки типів виконання для пропсів у додатках React. За допомогою PropTypes ви можете визначити типи пропсів, очікувані в компоненті, та, про те, чи є пропса обов'язковою. Коли ви передаєте інший тип до компонента або пропустите необхідну пропсу, ви отримаєте попередження в IDE, а потім і в консолі JavaScript.

PropTypes має дуже багато перевірок, і на кожну з них унікальне повідомлення у разі помилки. PropTypes вміє не тільки перевіряти масив прокинули чи об’єкт, але також і перевіряти із чого вони складаються. Також якщо пропса не є обов’язковою тоді треба назначити їй значення за замовчуванням (DefaultProp). Наприклад компонент галереї товарів має отримувати масив з товарами, і кожен товар повинен мати назву, ідентифікатор, ціну, валюту та зображення.

Gallery.propTypes = {

items: PropTypes.arrayOf(PropTypes.shape({

id: PropTypes.string.isRequired,

name: PropTypes.string.isRequired,

cost: PropTypes.number.isRequired,

currency: PropTypes.oneOf(['UAH', 'USD', 'EUR', 'RUB']).isRequired,

image: PropTypes.string,

})).isRequired,

}

І додаймо значення за замовчуванням для поля image, на випадок якщо її не буде, або десь загубиться по дорозі зі серверу.

Gallery.defaultProps = {

items: [{

image: "noImage.png"

}],

}

У підсумку PropTypes це невеличка бібліотека, яка робить корисну справу: допомагає уникнути помилки пов’язані із типом пропсів, а також дозволяє не пам’ятати всіх їх напам’ять.

Що таке TypeScript?

Оскільки складність сучасних веб-додатків порівнянна з настільними додатками, TypeScript в першу чергу призначений для виявлення помилок під час компіляції, а не під час виконання. Крім того, завдяки системі типів розробники отримують такі функції, як підказки і переходи коду, що значно прискорює процес розробки. Крім того, система типів в значній мірі позбавляє розробників від коментування коду, що займає багато часу. Також система типів допомагає виявити архітектурні проблеми раніше, корекція яких дешевше на ранніх стадіях.

Хоча на даний момент практично неможливо знайти бібліотеку, яка не була б перенесена в TypeScript, код JavaScript, що залишився від попередніх проектів, не повинен бути списаний, оскільки його спільний доступ не викличе ніяких проблем. Компілятор TypeScript відмінно справляється з обробкою динамічного коду JavaScript, включеного в його друковане середовище, і навіть виявляє помилки в ньому. Крім того, при компіляції файлів .ts в .js додатково генеруються файли декларацій .d.ts,за допомогою яких розробники, які створюють свої програми виключно на JavaScript, отримають всі ключові кнопки набраного автозаповнення в будь-якому сучасному середовищі розробки.

TypeScript значно скорочує час для виправлення помилок і виявлення помилок, які іноді не так просто знайти в динамічному середовищі JavaScript.

Якщо TypeScript є першою друкованою мовою для розробника, ви повинні знати, що навчання значно прискорить процес професійного зростання, оскільки типізований світ відкриває аспекти програмування, які не є очевидними в динамічних мовах.

Крім того, TypeScript дозволяє писати більш зрозумілий і читабельний код, який максимально описує предметну область, завдяки чому архітектура додатків стає більш вираженою, а розробка неявно підвищує професійний рівень програміста.

Все це, в повному обсязі, скорочує час розробки програми, знижуючи її вартість і надаючи розробникам можливість швидко приступити до реалізації нового і ще більш цікавого проекту.

* 1. CSS-in-JS

Бібліотеки CSS-in-JS надають нам новий підхід до написання CSS. Вони спрямовані на вирішення обмежень CSS, таких як відсутність динамічної функціональності, дослідження та портативності. Незважаючи на свої переваги, CSS-in-JS є спірною технологією, оскільки багато розробників запитують, чи варто ще більше ускладнюючи розробку front-end.

Бібліотеки CSS-in-JS набирають обертів з тих пір, як в зовнішній розробці з'явилися компонентні javascript фреймворки. Angular, React, Vue та інші фреймворки засновані на модулях, званих «компонентами», з яких ви можете створити цілу односторінкову програму (SPA). Компонент зазвичай є елементом інтерфейсу користувача, таким як кнопка, спливаюче вікно або панель переходів. Вам потрібно лише один раз створити компонент, і ви можете повторно використовувати його в будь-якому контексті в додатку.

Але, як ви повинні ефективно стилізувати SPA компоненти? Якщо ви використовуєте глобальні файли CSS, то буде важко сказати, як буде виглядати кінцевий результат. У зв'язку з каскадною природою CSS (Каскадні таблиці стилів), таблиці стилів можуть завантажуватися в будь-якому порядку і перевизначити один одного в будь-якій комбінації. Управління залежностями є ще однією проблемою, коли мова йде про стилізацію SPAs. Якщо управління залежністю важко при роботі зі звичайними веб-сайтами, це буде майже неможливо з модульним веб-додатком вище певної складності.

Добре розроблена бібліотека CSS-in-JS може вирішити всі ці проблеми. CSS-in-JS - це, по суті, бібліотека JavaScript, яка об'єднує кожен компонент JavaScript з усіма його правилами та залежностями CSS. В результаті компоненти можуть працювати самостійно, не покладаючись на будь-який зовнішній css-файл.

Переваги CSS-in-JS

Тепер давайте поговоримо про переваги бібліотек CSS-in-JS (хоча я вже згадував деякі з них).

* Локальне дослідження

За замовчуванням CSS не дозволяє локальне дослідження. Кожне правило стилю має глобальну сферу застосування, тому воно стосується всього проекту. В результаті стилі можуть перевизначити один одного дивним чином. Front-end розробники створили кілька методологій, які додають модульність CSS, такі як BEM, OOCSS і SMACSS. Попередні процесори та PostCSS - це ще один спосіб модульності CSS. Бібліотеки CSS-in-JS виводжуть цю концепцію на наступний рівень, автоматизуючи дослідження, що призводить до високого рівня передбачуваності.

* Інкапсуляція

Інкапсульовані компоненти приховують деталі реалізації. Вони лише піддають API зовнішньому світу, щоб інші компоненти могли взаємодіяти з ними. Інкапсуляція полегшує обслуговування та усуває помилки, оскільки ви можете змінювати весь код, пов'язаний з компонентом, в одному місці, не турбуючись про несподівану зміну інших частин програми.

* Портативність

Оскільки компоненти включають весь вихідний код, стилі та логіку, необхідні для правильного запуску, ви можете безпечно переміщати їх. Вони в основному працюють з коробки і дозволяють створювати вільно пов'язані програми, в яких компоненти спілкуються один з одним виключно за допомогою API.

* Повторне використання

Компоненти багаторазові, тому вам потрібно написати їх лише один раз, тоді ви можете запустити їх скрізь. Ви можете використовувати не тільки компоненти в одному додатку, але і в інших додатках, побудованих з тією ж структурою.

* Динамічна функціональність

Оскільки CSS-in-JS по суті є кодом JavaScript, ви можете застосувати складну логіку до правил стилю, таких як цикли, умовні, змінні, стиль на основі стану тощо. Тому це ідеальне рішення, якщо вам потрібно створити складний інтерфейс, який вимагає динамічної функціональності.

* Наслідування

Стилі можуть бути написані не тільки для нативних HTML-елементів, а також для вже існуючих компонентів зі своїми стилями.

Недоліки CSS-in-JS

Хоча бібліотеки CSS-in-JS дозволяють створювати програми на основі компонентів логічно та ефективно, вони також мають деякі характеристики, які можуть змусити вас побоютися їх.

* Крива навчання

CSS-in-JS, безумовно, має криву навчання, особливо якщо ви раніше не використовували ні компонентних фреймворків, ні веб-компонентів. Крім вивчення нового синтаксису, вам також потрібно підібрати новий спосіб мислення, який потребує часу і може уповільнити робочий процес розробки на деякий час.

* Додатковий рівень складності

Використання бібліотеки CSS-in-JS додає додатковий шар до вашого переднього стека, що іноді може бути непотрібним. Хоча CSS-in-JS чудово управляє складністю, це не завжди варто клопоту, особливо у випадку простішого проекту.

* Читабельність коду

Автоматично згенеровані селектори значно погіршують читабельність коду. Це може бути величезною проблемою для вас, якщо ви регулярно використовуєте інструменти розробника вашого браузера для налагодження, а також для початківців, оскільки для них важливо зрозуміти код, який вони написали.

Отже, коли використовувати CSS-in-JS бібліотеки?

Бібліотеки CSS-in-JS надають просту та безпечну методологію для створення компонентних додатків. Крім javascript фреймворків, ви також можете використовувати їх разом з веб-компонентами. Вони також можуть стати в нагоді, якщо ви хочете створити складний UX для монолітного інтерфейсу, такого як функціональність на основі держави.

* 1. Storybook

При розробці складних інтерфейсів продукту виникає питання подальшого масштабування і обслуговування кодової бази веб-сервісу. Згодом кількість компонентів інтерфейсу користувача зростає, поряд з цим збільшується поріг для вступу в проект - його розширення і налагодження утруднюються.

Навести думки і структуру проекту в порядок допоможе Storybook. Цей інструмент розробки інтерфейсу дозволяє реалізувати окремі частини інтерфейсу в ізоляції від основного додатку, що спрощує тестування і дозволяє налагоджувати компонент з різними даними. Для інтерактивної взаємодії з візуальними компонентами Storybook надає спеціальний веб-інтерфейс.

Однією з ключових особливостей Storybook є те, що він може бути використаний практично з будь-яким інструментарієм, будь то React.js, Vue.js або Angular. Є також багато плагінів, які розширюють його інтерфейс і додають додаткові методи роботи з компонентами.

Основними об'єктами взаємодії в Storybook є історії. Вони з'єднують середовище розробки і компоненти. Щоб відрізнити звичайні файли від історій, використовується ім'я .stories.js. Це правило, як і будь-яке інше, можна змінити, розширивши конфігурацію інструменту.

Після створення історії вона з'являється в інтерфейсі користувача разом з станом компонента. .stories.js файли включають серію історій, де кожен з них є функцією візуалізації, яка відображає стан компонента. Вважається хорошим чином, якщо один файл історії описує один і той же компонент в різних станах.

* 1. Firebase

Firebase - це платформа для розробки web та мобільних додатків з величезним функціоналом. Він починався як стартап, а сьогодні використовується при розробці кращих кросплатформних додатків.

Головною перевагою платформи є те, що вона дозволяє розробнику не відволікатися на створення бекенда, тобто прихованої від користувача програмної частини проекту, наприклад, коду сервера. А це спрощує і прискорює створення додатків, дає можливість повністю зосередитися на UI/UX, тобто на користувальницькому інтерфейсі і досвіді.

Firebase є одним з рішень BaaS (Backend as a Service), що дає розробнику масу можливостей. Це і сервер, і база даних, і хостинг, і аутентифікація в одній платформі. Наприклад, Firebase Realtime Database надає розробникам API, який синхронізує дані програми між клієнтами та зберігає їх у хмарному сховищі. Програма підключається до бази даних через WebSocket, який відповідає за синхронізацію даних протягом усього сеансу. Тобто клієнт не повинен кожну секунду відправляти запит на сервер, він сам сповістить про зміни. Це є найкращім рішенням для месенджерів.

Firebase також діє як сховище файлів. Firebase Storage забезпечує надійне завантаження та завантаження файлів для вашої програми. Хмарне сховище відео, аудіо або будь-якого іншого типу файлів підтримується Хмарним сховищем Google. Вміст хмарного сховища надійно захищений власною системою безпеки.

При створенні будь-якого додатку велика увага приділяється питанням безпеки. Створення системи аутентифікації кожен раз з нуля коштує досить дорого, і ці витрати найчастіше не виправдані. Впоратися з більшістю викликів дозволяє система аутентифікації Firebase Auth, в якій можна автентифікувати користувача програми паролем і електронною поштою. Firebase Auth також підтримує відкритий протокол авторизації OAuth 2.0, який використовується Google, Twitter, Facebook.

Система автентифікації Firebase інтегрується безпосередньо в базу даних. Статичні файли додатків розміщуються Firebase. Підтримується хостинг JavaScript, HTML, CSS та інших файлів. Динамічна підтримка Node.js реалізується через хмарні функції. Передача файлів здійснюється через мережу доставки контенту з використанням захищених протоколів SSL і HTTPS. Кросплатформне рішення Firebase Messaging дозволяє відправляти повідомлення на пристрої користувачів.

Сповіщення можуть надсилатися на будь-який тип пристрою, включаючи ПК – як індивідуальний, так і на груповий, або на всі пристрої, на яких встановлена програма. Рішення також підтримує можливість відправляти повідомлення навіть в окремих темах. Рішення легко масштабується і дозволяє відправляти величезну кількість повідомлень в найкоротші терміни або налаштовувати відправку повідомлень, наприклад, з урахуванням часового поясу одержувача.

1. ОПИС ПРОГРАМНОГО ПРОДУКТУ
   1. Заголовок першого підрозділу у третьому розділі

Текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст

* 1. Заголовок другого підрозділу у третьому розділі

Текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст текст

ВИСНОВКИ

Текст висновків Текст висновків Текст висновків Текст висновків Текст висновків Текст висновків Текст висновків Текст висновків Текст висновків Текст висновків Текст висновків

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

* 1. Javascript фреймворки. [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://maxsite.org/page/javascript-framework.
  2. Куріченко А. Javascript фреймворки [Електронний ресурс] / Олексій Куріченко. – 2015. – Режим доступу до ресурсу: https://www.reclamare.ua/blog/javascript-frejmvorki/.
  3. Single Page Application: як працює сайт-застосунок [Електронний ресурс]. – 2020. – Режим доступу до ресурсу: https://thecode.media/spa/.
  4. Що таке SPA або односторінковий портал? [Електронний ресурс] – Режим доступу до ресурсу: http://www.codenet.ru/webmast/js/spa/.
  5. Що таке SPA-додатки? [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://wezom.com.ua/blog/chto-takoe-spa-prilozheniya.
  6. Морозова Я. Что такое SPA, MPA и PWA [Електронний ресурс] / Яна Морозова. – 2021. – Режим доступу до ресурсу: https://www.azoft.ru/blog/spa-mpa-pwa/.
  7. Флэнаган Д. JavaScript: Полное руководство / Дэвид Флэнаган., 2021. – (7). – (722).
  8. Хавербеке М. Выразительный JavaScript / Марейн Хавербеке.. – 480 с.
  9. Кайл С. ES6 и не только / Симпсон Кайл., 2018. – 336 с.
  10. Пауэлл Д. Single Page Web Applications: JavaScript End-to-end / Д. Пауэлл, М. Миковский., 2013. – 432 с
  11. Мусиенко Ю. ANGULAR VS REACT VS VUE: ЛУЧШИЙ ВЫБОР В 2021 [Електронний ресурс] / Юрій Мусиенко. – 2021. – Режим доступу до ресурсу: https://merehead.com/ru/blog/angular-vs-react-vs-vue-2021/.
  12. Facebook. A javaScript Library for Building User Interfaces [Електронний ресурс] / Facebook. – 2017. – Режим доступу до ресурсу: https://facebook.github.io/React/.Accessed February 5 2017
  13. Stein J. ReactJS Cookbook / Johannes Stein., 2017. – 301 с. – (Kindle Edition).
  14. Why is React's concept of Virtual DOM said to be more performant than dirty model checking? [Електронний ресурс]. – 2014. – Режим доступу до ресурсу: http://stackoverflow.com/questions/21109361/why-is-Reacts-concept-ofvirtual-dom-said-to-be-more-performant-than-dirty-mode.
  15. [5.6 Маршрутизация (learn-reactjs.ru)](https://learn-reactjs.ru/training-project/routing)
  16. Abramov D. Getting Started with Redux [Електронний ресурс] / Dan Abramov. – 2021. – Режим доступу до ресурсу: https://redux.js.org/introduction/getting-started.
  17. Tilley M. What is Flux [Електронний ресурс] / Michelle Tilley. – 2014. – Режим доступу до ресурсу: http://fluxxor.com/what-is-flux.html.
  18. Flux и Redux [Електронний ресурс] // Metanit. – 2021. – Режим доступу до ресурсу: https://metanit.com/web/react/5.1.php.
  19. Flux-архитектура [Електронний ресурс] // Википедия. – 2019. – Режим доступу до ресурсу: https://ru.wikipedia.org/wiki/Flux-архитектура
  20. Catalin V. Изучаем архитектуру Flux в React [Електронний ресурс] / Vasile Catalin. – 2017. – Режим доступу до ресурсу: https://code.tutsplus.com/ru/tutorials/getting-started-with-the-flux-architecture-in-react--cms-28906.
  21. Yangshun T. In-Depth Overview [Електронний ресурс] / Tay Yangshun. – 2019. – Режим доступу до ресурсу: https://facebook.github.io/flux/docs/in-depth-overview.
  22. Бодух А. Flux Architecture / Адам Бодух., 2016. – 352 с.
  23. Garreau M. Redux in Action / M. Garreau, W. Faurot., 2018. – 312 с.
  24. [Проверка типов с помощью PropTypes – React (reactjs.org)](https://ru.reactjs.org/docs/typechecking-with-proptypes.html)
  25. [Що таке TypeScript і що це таке для | Javascript з прикладами коду (scriptdev.ru)](https://scriptdev.ru/ts/000/)
  26. [Вступ до CSS-in-JS: приклади, плюси та мінуси (tutsplus.com)](https://webdesign.tutsplus.com/articles/an-introduction-to-css-in-js-examples-pros-and-cons--cms-33574)
  27. [Зачем нужен инструмент Storybook и как он помогает нам жить (spark.ru)](https://spark.ru/startup/stud-olega-chulakova/blog/64850/zachem-nuzhen-instrument-storybook-i-kak-on-pomogaet-nam-zhit)

ЗАСТОСУНОК А. Назва додатку

Текст додатку

Текст додатку

Текст додатку

Текст додатку