Engineer

Фронтенд

частина системи, відповідальна за представлення й взаємодію

Що це таке

Фронтенд — це частина програмної системи, яку люди бачать і якою керують: сторінки, екрани, форми й елементи керування разом із кодом, що їх малює й реагує на введення. Англійською його також пишуть front end або front-end, а у вебі ту саму територію часто називають клієнтською стороною, client side.

У вебі фронтенд складається з трьох мов, які розуміє браузер. HTML задає сторінці структуру й зміст, CSS визначає її вигляд, а JavaScript додає поведінку. Браузер — це те, що документ W3C Architecture of the World Wide Web називає user agent: програма, яка діє від імені людини, отримує дані із серверів і показує їх. Усе, що фронтенд надсилає браузеру, опиняється на пристрої, який належить комусь іншому.

Його партнер — бекенд, який зберігає дані, застосовує бізнес-правила й відповідає на запити. Вони спілкуються через HTTP, зазвичай за допомогою API. Межу між ними хочеться провести там, де проходить мережа, але сучасні фреймворки її розмивають. У Next.js сторінки й шаблони компонування (layouts) за замовчуванням є Server Components: вони створюють те, що бачить користувач, і виконуються на сервері. Директива `"use client"` на початку файлу позначає межу, і цей файл, його імпорти та компоненти, які він рендерить, потрапляють у бандл, що вирушає до браузера. Фронтендом код робить його завдання, представлення й взаємодія, а місце виконання — окреме рішення.

Застосуйте поняття до рішення

Застосуйте «Що таке фронтенд?» до конкретного рішення. Ми можемо пов'язати поняття із системою, яку ви створюєте або купуєте. Інтерфейси, що швидко завантажуються, працюють на телефоні й поводяться однаково в усіх браузерах ваших клієнтів.

Переглянути наші послуги Зв'язатися з нами →

Перевірте припущення за терміном

Використайте пояснення та пов'язані роботи, щоб сформулювати запитання для проєкту. Ми можемо перевірити припущення разом до вибору інструмента. Керувала full-stack-розробкою GIS-карт — full-stack-картографія — від бази даних до браузера

Керувала full-stack-розробкою GIS-карт Зв'язатися з нами →

Чому це важливо

Перше рішення стосується того, де відбувається робота. Архітектурний документ W3C описує спектр, на одному кінці якого все робить клієнт, а на іншому все робить сервер, і зазначає, що більшість рішень поєднує обидва підходи. Місце застосунку на цьому спектрі залежить, за словами документа, від потужності клієнта, потужності й навантаження сервера та пропускної здатності між ними. Документ називає компроміс з обох боків: підлаштування вмісту на сервері знижує ефективність кешування, а коли кількість можливих клієнтів необмежена, краще масштабується перенесення обчислень на них. Звичні варіанти такі: сторінки, зібрані один раз, сторінки, які сервер рендерить на кожен запит, сторінки, складені в браузері, і їхні поєднання. Сторінку, однакову для всіх, найдешевше зібрати один раз і кешувати. Сторінку з даними конкретного користувача десь рендерять на кожен запит, і якщо це браузер, кожен читач завантажує код, що виконує цю роботу.

Друге рішення стосується довіри, і робоча відповідь така: сервер вважає все, що надходить від фронтенду, неперевіреним. Усе, що доставляє фронтенд, людина за браузером може прочитати, змінити й надіслати повторно. Прихована кнопка чи форма, яка відмовляється відправлятися, — це ввічливість щодо чесного користувача й невеликий обхід для будь-кого іншого, тому кожен дозвіл і кожна перевірка, які мають значення, повторюються на сервері. Секрети підкоряються тому самому правилу: API-ключ, що потрапив у бандл, є публічним, хоч би як називалася змінна. Content Security Policy обмежує, що може зробити впроваджений скрипт, а її специфікація подає її як багаторівневий захист, який доповнює ретельну валідацію введення й кодування виводу, а не замінює їх.

Фронтенд — це також місце, де доступність виграють або програють. WCAG 2.2, рекомендація W3C від грудня 2024 року, встановлює критерії успіху, які можна перевірити, і багато з них залежать від рішень у фронтенді: чи можна дістатися до елемента керування й скористатися ним із клавіатури та чи повідомляє він допоміжним технологіям своє ім’я й роль. Нативні елементи HTML виконують кілька з цих критеріїв одразу, а власний віджет мусить відтворити кожен вручну.

Нарешті, фронтенд, який знає, як база даних зберігає дані, доводиться змінювати щоразу, коли змінюється база, а фронтенд, що обмежується представленням, дає кожній стороні рухатися у власному темпі.

Як це виглядає на практиці

Фронтенд на Next.js 16 обирає режим рендерингу для кожної частини застосунку. Маркетингові й публічні сторінки генеруються статично, а інтерактивні частини рендеряться в браузері. На автентифікованому дашборді й великих сторінках-каталогах, де патерн N+1 перетворював одну сторінку на каскад мережевих звернень, фабрика `createPrefetchedServerPage` отримує дані сторінки на сервері й передає їх клієнту вже заповненими, а стратегію інвалідації кешу задокументовано для кожного запиту. Сторінка, перенесена на фабрику, завантажується за одне звернення, і наступна перенесена сторінка успадковує цю поведінку. Там же сказано, що перехід триває: решта автентифікованого застосунку досі рендериться на клієнті й стоїть у черзі.

У межах архітектурного внеску засновниці компанії у проєкт NextMariner — морську платформу — фронтенду відведено одне завдання й нічого більше. Фронтенд на Next.js відповідає за відображення та інтерактивність. Навмисно тонкий Go API, побудований на Huma поверх Fiber, маршрутизує й перевіряє запити, застосовує фільтрацію з міркувань безпеки та оркеструє, а бізнес-логіка живе у функціях PostgreSQL. Схема зосереджує зміну бізнес-правила в одному шарі, у SQL, замість того щоб вимагати те саме правило в усіх трьох. Межі записали, а потім забезпечили їх дотримання під час рев'ю коду. Зафіксований результат — розділена відповідальність за відображення, обробку запитів і бізнес-правила.

Робота з посилення захисту застосунку виходить із припущення, що клієнт ворожий. У фронтенді проксі-мідлвар Next.js у `proxy.ts` задає Content Security Policy із nonce, згенерованим для кожного запиту, і `strict-dynamic`, а також HSTS і cookie з атрибутом SameSite. Сама межа проходить на сервері. Права доступу, тобто тариф, роль, організацію й прив'язку до судна, сервер повторно перевіряє на кожен запит, а фронтендні обмеження розглядаються лише як UX: вони визначають, що бачить людина, а сервер визначає, що їй дозволено робити.

Де виникають проблеми

Збої фронтенду зазвичай мають спільну причину: щось вирішили в браузері, хоча це належало серверу, або навпаки.

API-ключ з'являється в JavaScript, який може завантажити будь-який відвідувач. У Next.js змінна середовища потрапляє в бандл браузера, лише коли її назва починається з `NEXT_PUBLIC_`, а будь-яку іншу змінну, прочитану з клієнтського коду, замінює порожній рядок. Запит повертається неавторизованим, хтось швидко перейменовує змінну, щоб усе запрацювало, і секрет опиняється в публічному файлі. Код, якому потрібен секрет, варто тримати на сервері, а його модуль позначати пакетом `server-only`, щоб імпорт із Client Component зупиняв збирання. Кожен ключ, який уже потрапив у реліз, треба замінити.

```
import "server-only";
export async function getInvoices(customerId: string) {
    const url = `${process.env.BILLING_URL}/customers/${customerId}/invoices`;
    const headers = { Authorization: `Bearer ${process.env.BILLING_KEY}` };
    const res = await fetch(url, { headers });
    return res.json();
}
```

Користувач дістається записів, які інтерфейс від нього приховав. Перевірка дозволів жила у фронтенді як route guard, прихований пункт меню або вимкнена кнопка. Будь-хто, хто читає мережевий трафік, може викликати API напряму з іншими параметрами. Кожне право варто перевіряти на сервері на кожен запит, щодо користувача, який увійшов, і конкретного запису, а перевірки у фронтенді вважати зручністю для чесних користувачів.

Сторінка заповнюється по одному спінеру за раз. Кожен компонент отримує власні дані після рендерингу, тож запит дочірнього компонента чекає на запит батьківського, і сторінка приходить водоспадом мережевих звернень. Дані сторінки варто отримувати разом і ближче до джерела, на сервері, де це дозволяє фреймворк, і передавати вниз.

Користувачі клавіатури й екранних читачів застрягають на саморобному елементі керування. Клікабельний `div` отримує поведінку кнопки для миші й більше нічого: клавіша Tab його оминає, Enter і пробіл нічого не

роблять, а допоміжні технології не дізнаються, чим він є. WCAG 2.2 на рівні A вимагає, щоб уся функціональність працювала з клавіатури (критерій успіху 2.1.1) і щоб кожен компонент повідомляв своє ім'я та роль (4.1.2). Варто використовувати `button`, `a`, `input` і `label` за їхнім призначенням і пробувати кожен новий екран лише з клавіатурою, перш ніж він вийде в реліз.

Що почитати далі

- Architecture of the World Wide Web, Volume One — рекомендація W3C про user agents і про розподіл роботи між клієнтом і сервером.
- HTML Standard — живий стандарт WHATWG для елементів і браузерних API, з яких будують вебфронтенд.
- Web Content Accessibility Guidelines (WCAG) 2.2 — рекомендація W3C із критеріями успіху, на яких тримається доступний інтерфейс.
- Content Security Policy Level 3 — робоча чернетка W3C, яка визначає nonces і `strict-dynamic`.
- Next.js: Server and Client Components — як один фреймворк розподіляє код фронтенду між сервером і браузером.

Словник

Обирайте за фактичними умовами

Опишіть робочий процес, наявні інтерфейси та користувачів, яким потрібна зміна. Ми можемо визначити перевірюваний результат і спосіб передачі, який дасть вашій команді змогу підтримувати його.

Дивитися, скільки це коштує Зв'язатися з нами →

Зробіть термінологію практичною

Запитайте, як «Що таке фронтенд?» стосується вашої організації. Конкретного прикладу з поточної системи достатньо для початку розмови.

Переглянути наші послуги

welcome@engineer.company +45 50 10 61 84 Копенгаген, Данія

Візьміть цю сторінку із собою

Візитівка

Дбайливо зроблено з любов'ю — D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Авторське право © 2025 – дотепер · Engineer ApS · Усі права захищено.

Надруковано з https://engineer.company/uk/glossary/frontend/