Engineer

Frontend

der Systemteil für Darstellung und Bedienung

Was es ist

Ein Frontend ist der Teil eines Softwaresystems, den Menschen ansehen und bedienen: Seiten, Bildschirme, Formulare und Bedienelemente samt dem Code, der sie zeichnet und auf Eingaben reagiert. Im Englischen gibt es auch die Schreibweisen front end und front-end. Im Web wird derselbe Bereich oft Clientseite genannt.

Im Web entsteht ein Frontend aus drei Sprachen, die ein Browser versteht. HTML gibt einer Seite Struktur und Bedeutung, CSS bestimmt ihr Aussehen, und JavaScript ergänzt Verhalten. Der Browser ist das, was die W3C-Schrift Architecture of the World Wide Web einen User Agent nennt: Software, die im Auftrag einer Person Daten von Servern abruft und darstellt. Alles, was ein Frontend an ihn sendet, landet auf einem fremden Gerät.

Sein Gegenstück ist das Backend. Es speichert Daten, wendet Geschäftsregeln an und beantwortet Anfragen. Beide sprechen über HTTP, meist über eine API. Die Grenze entlang des Netzwerks zu ziehen liegt nahe, doch heutige Frameworks verwischen sie. In Next.js sind Seiten und Layouts standardmäßig Server Components: Sie erzeugen das Sichtbare, laufen aber auf einem Server. Die Direktive `"use client"` am Dateianfang markiert die Grenze. Diese Datei, ihre Imports und die von ihr gerenderten Komponenten werden Teil des an den Browser gesendeten Bundles. Code gehört wegen seiner Aufgabe zum Frontend: Darstellung und Interaktion. Sein Ausführungsort ist eine eigene Entscheidung.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist ein Frontend?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Oberflächen, die schnell laden, auf dem Smartphone funktionieren und sich in allen Browsern Ihrer Kunden gleich verhalten.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Full-Stack-GIS-Kartenentwicklung geleitet — Full-Stack-Kartenentwicklung von der Datenbank bis zum Browser

Full-Stack-GIS-Kartenentwicklung geleitet Kontakt aufnehmen →

Warum es wichtig ist

Die erste Entscheidung ist, wo die Arbeit stattfindet. Das W3C-Architekturdokument beschreibt ein Spektrum: An einem Ende erledigt der Client alles, am anderen der Server. Die meisten Entwürfe verbinden laut Dokument beides. Wo eine Anwendung einzuordnen ist, hängt in dessen Worten von der Leistungsfähigkeit des Clients, der Leistung und Last des Servers sowie der Bandbreite dazwischen ab. Es benennt den Zielkonflikt von beiden Seiten: Serverseitig angepasste Inhalte senken die Cache-Effizienz. Ist die mögliche Clientzahl unbegrenzt, skaliert die Verlagerung der Berechnung auf die Clients besser. Üblich sind einmal gebaute Seiten, pro Anfrage auf dem Server gerenderte Seiten, im Browser zusammengesetzte Seiten und Mischformen. Eine für alle gleiche Seite wird am günstigsten einmal gebaut und gecacht. Eine Seite mit den Daten eines einzelnen Kunden wird irgendwo pro Anfrage gerendert. Fällt die Wahl auf den Browser, lädt jeder Leser den dafür nötigen Code herunter.

Die zweite Entscheidung ist, wie weit ihm vertraut wird. Die praktische Antwort lautet: Der Server vertraut ihm überhaupt nicht. Alles vom Frontend Ausgelieferte kann die Person am Browser lesen, bearbeiten und erneut abspielen. Ein versteckter Button oder ein Formular, das sich nicht absenden lässt, hilft ehrlichen Nutzern und bedeutet für andere nur einen kleinen Umweg. Jede wichtige Berechtigungs- und Validierungsprüfung wird deshalb auf dem Server wiederholt. Für Geheimnisse gilt dasselbe: Ein API-Schlüssel im Bundle ist öffentlich, unabhängig vom Variablennamen. Content Security Policy begrenzt, was ein eingeschleustes Skript tun kann. Ihre Spezifikation beschreibt sie als zusätzliche Verteidigungsschicht, nie als Ersatz für sorgfältige Eingabeprüfung und Ausgabekodierung.

Im Frontend entscheidet sich auch die Barrierefreiheit. WCAG 2.2, die im Dezember 2024 veröffentlichte W3C-Empfehlung, legt prüfbare Erfolgskriterien fest. Viele hängen von Frontend-Entscheidungen ab: Ist ein Bedienelement per Tastatur erreichbar und bedienbar, und stellt es assistiven Technologien einen Namen und eine Rolle bereit? Native HTML-Elemente erfüllen mehrere davon von Haus aus. Ein eigenes Widget muss jedes von Hand nachbauen.

Schließlich muss sich ein Frontend, das die Speicherstruktur der Datenbank kennt, bei jeder Datenbankänderung mitändern. Beschränkt es sich auf Darstellung, können beide Seiten sich dagegen im eigenen Tempo weiterentwickeln.

Wie es in der Praxis aussieht

Auf einer Plattform wird Teil für Teil entschieden, wo das Frontend rendert. Ihr Frontend mit Next.js 16 wählt für jeden Anwendungsteil einen Rendering-Modus. Marketing- und öffentliche Seiten werden statisch erzeugt, interaktive Teile im Browser gerendert. Im Dashboard für angemeldete Nutzer und in großen Verzeichnislisten machte das N+1-Muster aus einer Seite eine Kaskade von Netzwerkumläufen. Dort lädt eine Factory namens `createPrefetchedServerPage` die Seitendaten auf dem Server vor und übergibt sie anschließend dem Client. Für jede Datenanfrage ist eine Strategie zur Cache-Invalidierung dokumentiert. Eine auf die Factory umgestellte Seite lädt in einem Netzwerkumlauf; die nächste umgestellte Seite erbt dieses Verhalten. Die Migration läuft noch. Der übrige nur nach Anmeldung zugängliche Anwendungsteil wird auf dem Client gerendert und wartet auf die Umstellung.

Der Architekturbeitrag der Gründerin zu NextMariner, einer maritimen Plattform, gibt ihrem Frontend genau eine Aufgabe. Das Frontend in Next.js übernimmt Darstellung und Interaktivität. Eine bewusst dünne Go-API mit Huma auf Fiber routet und validiert Anfragen, wendet Sicherheitsfilter an und koordiniert die Abläufe. Die Geschäftslogik liegt dagegen in PostgreSQL-Funktionen. Eine Geschäftsregel wird in diesem Entwurf in einer Schicht geändert, in SQL, statt dieselbe Regel in allen drei Schichten vorhalten zu müssen. Die Grenzen wurden aufgeschrieben und anschließend in Code-Reviews durchgesetzt. Das dokumentierte Ergebnis sind getrennte Verantwortlichkeiten für Darstellung, Anfragebearbeitung und Geschäftsregeln.

Die Arbeit an der Anwendungshärtung geht von einem feindlichen Client aus. Auf Frontend-Seite setzt die Proxy-Middleware von Next.js in `proxy.ts` eine Content Security Policy mit einem pro Anfrage erzeugten Nonce und `strict-dynamic`, zusammen mit HSTS und SameSite-Cookies. Die eigentliche Grenze liegt auf dem Server. Berechtigungen für Tarifstufe, Rolle sowie die Abgrenzung nach Organisation und Schiff werden für jede Anfrage erneut auf dem Server geprüft. Die Zugangsschranken des Frontends dienen allein der Nutzerführung: Sie formen das Sichtbare, der Server entscheidet über das Erlaubte.

Wo es schiefgeht

Frontend-Fehler haben oft dieselbe Ursache: Im Browser wurde etwas entschieden, das auf den Server gehörte, oder umgekehrt.

Ein API-Schlüssel steht im JavaScript, das jeder Besucher herunterladen kann. In Next.js gelangt eine Umgebungsvariable nur dann ins Browser-Bundle, wenn ihr Name mit `NEXT_PUBLIC_` beginnt. Jede andere aus Clientcode gelesene Variable wird durch eine leere Zeichenfolge ersetzt. Kommt eine Anfrage als nicht autorisiert zurück und wird die Variable rasch umbenannt, damit es funktioniert, landet das Geheimnis direkt in einer öffentlichen Datei. Halten Sie Code mit benötigten Geheimnissen auf dem Server und markieren Sie sein Modul mit dem Paket `server-only`. Ein Import aus einer Client Component lässt dann den Build scheitern. Rotieren Sie jeden bereits ausgelieferten Schlüssel.

```
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();
}
```

Ein Nutzer erreicht Datensätze, die ihm die Oberfläche verborgen hatte. Die Berechtigungsprüfung lag im Frontend, als Routenschutz, versteckter Menüeintrag oder deaktivierter Button. Wer den Netzwerkverkehr liest, kann die API direkt mit anderen Parametern aufrufen. Prüfen Sie jede Berechtigung auf dem Server bei jeder Anfrage anhand des angemeldeten Nutzers und des konkreten Datensatzes. Behandeln Sie die Frontend-Prüfungen als Komfort für ehrliche Nutzer.

Die Seite füllt sich Ladeanzeige für Ladeanzeige. Jede Komponente fragt ihre Daten erst nach dem Rendern ab. Eine Anfrage des Kindelements wartet deshalb auf die des Elternelements. Die Seite kommt als Wasserfall von Netzwerkumläufen an. Laden Sie die Daten einer Seite gemeinsam und nahe ihrer Quelle, auf dem Server, wo das Framework es zulässt, und reichen Sie sie nach unten weiter.

Tastatur- und Screenreader-Nutzer bleiben an einem eigenen Bedienelement hängen. Ein anklickbares `div` bekommt das Mausverhalten eines Buttons, aber sonst nichts: keinen Fokus, keine Reaktion auf Enter und Leertaste, keine Auskunft über seine Art an assistive Technologien. WCAG 2.2 verlangt auf Stufe A, dass alle Funktionen per Tastatur bedienbar sind, Erfolgskriterium 2.1.1, und jede Komponente ihren Namen und ihre Rolle offenlegt, 4.1.2. Verwenden Sie `button`, `a`, `input` und `label` ihrer Bedeutung entsprechend. Probieren Sie jeden neuen Bildschirm vor der Auslieferung allein mit der Tastatur aus.

Weiterführende Lektüre

- Architektur des World Wide Web, Band Eins — die W3C-Empfehlung zu User Agents und zur Arbeitsteilung zwischen Client und Server.
- HTML-Standard — der fortlaufend gepflegte WHATWG-Standard für Elemente und Browser-APIs eines Web-Frontends.
- Richtlinien für barrierefreie Webinhalte (WCAG) 2.2 — die W3C-Empfehlung mit den prüfbaren Erfolgskriterien einer barrierefreien Oberfläche.
- Content Security Policy Level 3 — der W3C-Arbeitsentwurf, der Nonces und `strict-dynamic` definiert.
- Next.js: Server- und Client-Komponenten — wie ein Framework den Frontend-Code zwischen Server und Browser aufteilt.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist ein Frontend?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/frontend/