Engineer

フロントエンド

システムの表示と操作を担当する部分

どのようなものか

フロントエンドは、ソフトウェアシステムのうち、人が見て操作する部分です。ページ、画面、フォーム、操作部品と、それを描いて入力へ応じるコードを含みます。英語ではfront endやfront-endとも書き、ウェブでは、同じ領域をクライアント側ともよく呼びます。

ウェブでは、ブラウザーが理解する三つの言語で作ります。HTMLが構造と意味を、CSSが見た目を、JavaScriptが振る舞いを与えます。ブラウザーは、W3CのArchitecture of the World Wide Webが、利用者に代わってサーバーからデータを取得し表示するソフトウェア、ユーザーエージェントと呼ぶものです。フロントエンドが送るものはすべて、サービス提供者とは別の人が所有するデバイスへ届きます。

対になるのは、データを保存し、業務規則を適用し、要求へ答えるバックエンドです。通常はAPIを通じ、HTTPで通信します。ネットワークを境界にしたくなりますが、今のフレームワークは、それを曖昧にします。Next.jsでは、ページとレイアウトは、既定でサーバーコンポーネントです。利用者の見るものを作っても、実行するのはサーバーです。ファイル先頭の `"use client"` が境界を示し、そのファイル、インポートしたもの、描くコンポーネントが、ブラウザーへ送るバンドルに入ります。コードがフロントエンドになる理由は、表示と操作という役割です。実行場所は、別の判断です。

概念を判断に生かす

「フロントエンドとは？」を実際の判断に生かしましょう。その概念を、開発または購入するシステムに結び付けて考えるお手伝いをします。 読み込みが速く、スマートフォンでも使え、お客様がどのブラウザーを使っていても同じように動く画面を提供します。

サービスを見る お問い合わせ →

用語の背後にある前提を検証する

解説とリンク先の実績を参考に、プロジェクトについての質問を整理してください。ツールを選ぶ前に、前提を一緒に検討できます。 フルスタックのGIS地図開発を指揮 — DBからブラウザーまでのフルスタック地図開発

フルスタックのGIS地図開発を指揮 お問い合わせ →

なぜ重要なのか

最初に決めるのは、処理をどこで行うかです。W3Cのアーキテクチャー文書は、クライアントがすべてを処理する構成から、サーバーがすべてを処理する構成までの幅を示し、多くの設計が両者を組み合わせていると述べています。処理をどこに置くかは、同文書によれば、クライアントの能力、サーバーの能力と負荷、両者の間の帯域幅によって決まります。双方のトレードオフも説明します。サーバーで内容を個別化するとキャッシュの効率が下がり、クライアント数に上限がなければ、計算をそちらへ分散する方が、規模を広げやすくなります。一般的なのは、一度だけ作るページ、要求ごとにサーバーで描くページ、ブラウザーで組み立てるページ、その混合です。全員に同じなら、一度作ってキャッシュするのが最も安く、特定の顧客のデータなら、どこかで要求ごとに描きます。ブラウザーを選べば、読者全員が、そのコードを取得します。

二つ目は、どこまで信頼するかです。実際の答えは、サーバーは何も信頼しない、というものです。ブラウザーを持つ人は、届いたものを読み、変え、再送できます。隠したボタンや送信を拒むフォームは、正直な利用者への配慮で、ほかの人には、小さい回り道です。そのため、重要な権限と検証は、サーバーで繰り返します。秘密も同じです。バンドルへ届いたAPIキーは、変数名にかかわらず公開です。Content Security Policyは、注入したスクリプトのできることを狭めます。仕様も、多層防御とし、慎重な入力検証と出力の符号化を、決して置き換えないとします。

アクセシビリティーも、ここで決まります。2024年12月公開のW3C勧告、WCAG 2.2は、検査できる達成基準を定め、多くがフロントエンドの選択にかかります。キーボードで操作部品へ届いて使えるか、支援技術へ名前と役割を示すか、などです。標準のHTML要素はいくつかの要件を最初から満たしますが、独自のウィジェットでは、それぞれの要件を満たす動作を手作業で実装し直す必要があります。

そして、データベースの保存方法を知るフロントエンドは、データベースの変更のたびに、変わらなければなりません。表示に専念すれば、両側をそれぞれの都合に合わせて変更できます。

実際の活用例

あるプラットフォームのNext.js 16のフロントエンドは、アプリの部分ごとに、描画方法を選びます。マーケティングと公開のページは静的に生成し、操作する部分はブラウザーで描きます。ログイン後のダッシュボードや大規模なディレクトリー一覧では、N+1によって、一ページが往復通信の連鎖になっていました。`createPrefetchedServerPage` というファクトリーが、サーバーでデータを先に取り、取得済みの状態でクライアントへ渡します。各データ要求について、キャッシュの無効化方針も文書化しています。移したページは、一往復で読み込め、次に移行するページも、その動作を引き継ぎます。移行は進行中で、認証が必要なアプリの残りの部分は、クライアントで描画しながら移行の順番を待っています。

創業者による、海事プラットフォームNextMarinerでの設計への貢献は、フロントエンドの仕事を、一つだけにします。Next.jsのフロントエンドは、表示と操作を担当します。Fiber上のHumaで作る、責務を意図的に絞ったGoのAPIが、経路を決め、検証し、セキュリティフィルターを適用し、処理を調整します。業務ロジックは、PostgreSQLの関数に置きます。同じ規則を三層すべてに実装するのではなく、SQLという一層だけで業務規則を変更できる設計です。境界を文書化し、コードレビューで守りました。記録された結果は、表示、要求の処理、業務規則の責任が、分かれたことです。

アプリを強化する仕事は、クライアントを敵対的と見なすことから始まります。フロントエンドでは、Next.jsの `proxy.ts` のプロキシーミドルウェアが、要求ごとに生成したnonceと `strict-dynamic` を持つContent Security Policyを、HSTS、SameSite Cookieとともに設定します。境界自体は、サーバーです。利用プランの段階、役割、組織と船舶の範囲を含む権限を、サーバーで全要求に対して再確認します。フロントエンドの制限は、利用体験にすぎません。見えるものを形作り、できることは、サーバーが決めます。

問題が起きるところ

問題は、ブラウザーで決めたことがサーバーの仕事だった、または、その逆だった、という原因を共有しがちです。

全訪問者が取得できるJavaScriptへ、APIキーが出ます。 Next.jsでは、名前が `NEXT_PUBLIC_` で始まる環境変数だけが、ブラウザーのバンドルへ届きます。クライアントコードから読むほかの変数は、空の文字列になります。要求が認可されないからと、動かすために環境変数の名前を急いで変えると、公開ファイルへ秘密を埋め込みます。秘密を使うコードはサーバーへ置き、モジュールに `server-`

`only` パッケージを読み込ませ、クライアントコンポーネントからのインポートで、ビルドを失敗させます。すでに外部へ配信されたキーは、すべて交換します。

```
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();
}
```

画面が隠した記録へ、利用者が届きます。 権限の検査が、ルートガード、隠したメニュー、無効化されたボタンとして、フロントエンドにありました。通信を読む人は、別の引数でAPIを直接呼べます。要求ごとに、すべての権限を、ログイン中の利用者と該当する記録に照らして、サーバーで確かめます。画面側は、正直な利用者の利便性として扱います。

読み込み表示ごとに、少しずつ埋まります。 各コンポーネントが、描いた後でデータを取り、子コンポーネントの要求が親コンポーネントの要求を待つため、ページの読み込みが、往復通信の連鎖するウォーターフォールになります。一ページのデータをまとめてデータの取得元に近い場所で、フレームワークが対応している場合はサーバー側で取得し、子コンポーネントへ渡します。

独自の部品を、キーボードやスクリーンリーダーの利用者が操作できません。 クリックできる `div` は、マウスでボタンのように操作できても、ボタンに備わるほかの機能はありません。フォーカスを受けず、EnterとSpaceを無視し、支援技術へ種類も示しません。WCAG 2.2は、レベルAで、全機能のキーボード操作を2.1.1、全コンポーネントの名前と役割を4.1.2で求めます。 `button` 、 `a` 、 `input` 、 `label` を、本来の用途で使い、公開前に、新しい画面をすべて、キーボードだけで試します。

参考資料

- World Wide Webのアーキテクチャー、第一巻 — ユーザーエージェントと、クライアントとサーバーの分担についての、W3C勧告です。
- HTML標準 — ウェブのフロントエンドを作る、要素とブラウザーAPIについての、WHATWGのLiving Standardです。
- ウェブコンテンツのアクセシビリティー指針、WCAG 2.2 — 操作画面のアクセシビリティーの基となる、検査可能な達成基準を持つW3C勧告です。

- Content Security Policy Level 3 — nonceと `strict-dynamic` を定義する、W3Cの作業草案です。
- Next.js：サーバーとクライアントのコンポーネント — 一つのフレームワークが、フロントエンドのコードをサーバーとブラウザーへどう分担させるかを説明します。

用語集

実際の制約に合わせて選ぶ

業務の流れ、既存のインターフェース、変更を必要とする利用者をお知らせください。検証可能な納品内容と、お客様のチームが保守を続けられる引き継ぎ内容を定められます。

料金を見る お問い合わせ →

用語を実務に結び付ける

「フロントエンドとは？」がお客様の組織にどう当てはまるかをご相談ください。現在のシステムの具体例が一つあれば、話を始められます。

サービスを見る

welcome@engineer.company +45 50 10 61 84 デンマーク、コペンハーゲン

このページを持ち歩く

連絡先カード

心を込めて丁寧に制作： D.

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

著作権 © 2025 – 現在 · Engineer ApS · すべての権利を留保します。

印刷元： https://engineer.company/ja/glossary/frontend/