Engineer

Frontend

systemdelen som hanterar presentation och interaktion

Vad det är

En frontend är den del av ett mjukvarusystem som människor tittar på och använder: sidor, vyer, formulär och kontroller, plus koden som ritar upp dem och svarar på inmatning. Ordet skrivs också front end eller front-end, och på webben kallas samma område ofta klientsidan.

På webben byggs en frontend av tre språk som webbläsaren förstår. HTML ger sidan struktur och betydelse, CSS bestämmer hur den ser ut och JavaScript lägger till beteende. W3C:s dokument Architecture of the World Wide Web kallar webbläsaren en user agent, alltså mjukvara som agerar för en persons räkning, hämtar data från servrar och visar dem. Allt en frontend skickar dit hamnar därmed på en enhet som tillhör någon annan.

Motparten är backend, som lagrar data, tillämpar affärsreglerna och svarar på anrop. De två talar med varandra över HTTP, oftast genom ett API. Det ligger nära till hands att dra gränsen där nätverket går, men dagens ramverk suddar ut den. I Next.js är sidor och layouter Server Components som standard: de tar fram det användaren ser och körs på en server. Ett `"use client"` -direktiv överst i en fil markerar gränsen, och den filen, dess importer och komponenterna den renderar hamnar i bundlen som skickas till webbläsaren. Det som gör kod till frontend är uppgiften, presentation och interaktion. Var koden körs är ett separat beslut.

Använd begreppet i ett beslut

Använd ”Vad är en frontend?” i ett konkret beslut. Vi kan knyta begreppet till systemet ni bygger eller köper. Gränssnitt som laddar snabbt, fungerar i mobilen och beter sig likadant i alla webbläsare dina kunder råkar använda.

Se våra tjänster Hör av er →

Granska antagandena bakom ordet

Använd förklaringen och det länkade arbetet för att formulera frågor till projektet. Vi kan undersöka antagandena tillsammans innan ett verktyg väljs. Styrde full stack-utvecklingen av GIS-kartor — kartutveckling i full stack, från databas till webbläsare

Styrde full stack-utvecklingen av GIS-kartor Hör av er →

Varför det spelar roll

Det första beslutet gäller var arbetet görs. W3C:s arkitekturdokument beskriver en skala med en klient som gör allt i ena änden och en server som gör allt i den andra, och konstaterar att de flesta lösningar blandar de två. Var en applikation hör hemma beror enligt dokumentet på klientens kapacitet, serverns kapacitet och belastning och bandbredden mellan dem. Avvägningen beskrivs från båda håll: innehåll som skräddarsys på servern ger sämre effekt av cachning, och när antalet möjliga klienter saknar övre gräns skalar det bättre att flytta beräkningar ut till dem. I praktiken väljer man mellan sidor som byggs en gång, sidor som renderas på en server vid varje anrop, sidor som sätts ihop i webbläsaren och blandningar av dessa. En sida som är likadan för alla är billigast att bygga en gång och lägga i cache. En sida med en enskild kunds data måste renderas per anrop någonstans, och väljer man webbläsaren laddar varje läsare ner koden som gör det.

Det andra beslutet gäller hur mycket man litar på frontenden, och svaret som fungerar är att servern inte litar på något av det den får. Allt en frontend levererar kan läsas, ändras och spelas upp igen av den som håller i webbläsaren. En dold knapp eller ett formulär som vägrar skicka är en artighet mot en ärlig användare och en liten omväg för alla andra, så varje behörighet och varje validering som spelar roll görs om på servern. Hemligheter lyder under samma regel: en API-nyckel som når bundlen är publik, oavsett vad variabeln heter. Content Security Policy begränsar vad ett injicerat skript kan göra, och specifikationen beskriver den som ett extra försvarslager, aldrig som en ersättning för noggrann validering av indata och kodning av utdata.

Frontenden är också platsen där webbtillgänglighet vinns eller går förlorad. WCAG 2.2, den W3C-rekommendation som publicerades i december 2024, ställer upp testbara framgångskriterier, och många av dem avgörs av val i frontenden: om en kontroll går att nå och använda med tangentbordet, och om den talar om sitt namn och sin roll för hjälpmedel. Inbyggda HTML-element uppfyller flera av kriterierna direkt, medan en egenbyggd komponent måste återskapa vart och ett för hand.

Till sist en fråga om koppling. En frontend som vet hur databasen lagrar saker måste ändras varje gång databasen gör det, medan en frontend som håller sig till presentation låter båda sidor röra sig i sin egen takt.

Så ser det ut i praktiken

På tre sidor på den här webbplatsen har frontenden fått tydliga kanter: var den renderar, vad den gör och vad den betros med.

På en plattform väljer Next.js 16-frontenden renderingsläge för varje del av appen. Marknadsföringssidor och publika sidor genereras statiskt, och de interaktiva delarna renderas i webbläsaren. I dashboarden för inloggade användare och i de stora katalogsidorna gjorde N+1-mönstret en enda sida till en kaskad av rundresor. Där hämtar en fabrik, `createPrefetchedServerPage`, sidans data på servern och lämnar över dem till klienten redan ifyllda, och en strategi för cacheinvalidering är dokumenterad för varje datahämtning. En sida som har flyttats till fabriken laddas med en enda rundresa, och nästa sida som flyttas ärver samma beteende. Sidan beskriver det som en pågående migrering: resten av den inloggade appen renderas fortfarande på klienten och väntar på sin tur.

Grundarens arkitekturbidrag till NextMariner, en maritim plattform, ger frontenden ett enda jobb och inget annat. Frontenden i Next.js sköter presentation och interaktivitet. Ett medvetet tunt API i Go, byggt med Huma ovanpå Fiber, routar och validerar anrop, tillämpar säkerhetsfiltrering och orkestrerar, medan affärslogiken ligger i PostgreSQL-funktioner. Designen placerar en ändring av en affärsregel i ett lager, i SQL, i stället för att kräva samma regel i alla tre. Gränserna skrevs ned och upprätthölls sedan i kodgranskningen. Det dokumenterade resultatet är att presentation, hantering av anrop och affärsregler har separata ansvarsområden.

Arbetet med att härda en applikation utgår från att klienten är fientlig. I frontenden sätter middlewaren i Next.js, `proxy.ts`, en Content Security Policy med en nonce som genereras per anrop och `strict-dynamic`, tillsammans med HSTS och SameSite-cookies. Själva gränsen sitter ändå på servern. Allt som frontenden döljer eller spärrar räknas där som användarupplevelse, och servern går igenom nivå, roll, organisation och fartygsavgränsning en gång till för varje anrop. Frontenden styr vad en person ser, och servern avgör vad personen får göra.

Var det går fel

Felen i en frontend har ofta samma orsak: något avgjordes i webbläsaren som hörde hemma på servern, eller tvärtom.

En API-nyckel dyker upp i JavaScript som vilken besökare som helst kan ladda ner. I Next.js når en miljövariabel bundlen i webbläsaren bara om namnet börjar med `NEXT_PUBLIC_`, och varje annan variabel som klientkod läser ersätts med en tom sträng. Ett anrop som kommer tillbaka som obehörigt, följt av ett snabbt namnbyte för att få det att fungera, bäddar in hemligheten i en publik fil. Låt kod som behöver en hemlighet stanna på servern, och märk modulen med paketet `server-only`, så att bygget avbryts om en Client Component importerar den. Byt ut varje nyckel som redan har skickats ut.

```
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();
}
```

En användare når poster som gränssnittet hade gömt. Behörighetskontrollen låg i frontenden, som ett skydd på en route, ett dolt menyval eller en inaktiverad knapp. Den som läser nätverkstrafiken kan anropa API:et direkt med andra parametrar. Kontrollera varje rättighet på servern vid varje anrop, mot den inloggade användaren och den specifika posten, och behandla frontendens kontroller som en bekvämlighet för ärliga användare.

Sidan fylls i en spinner i taget. Varje komponent hämtar sina egna data först efter att den har renderats, så ett barns anrop väntar på förälderns och sidan kommer fram som ett vattenfall av rundresor. Hämta en sidas data samlat och nära källan, på servern där ramverket tillåter det, och skicka dem nedåt.

Den som använder tangentbord eller skärmläsare fastnar i en egenbyggd kontroll. En klickbar `div` får en knapps beteende för musen och inget av resten: den tar inte emot fokus, struntar i Enter och mellanslag och berättar ingenting för hjälpmedel om vad den är. WCAG 2.2 kräver på nivå A att all funktionalitet fungerar från tangentbordet (framgångskriterium 2.1.1) och att

varje komponent exponerar sitt namn och sin roll (4.1.2). Använd `button`, `a`, `input` och `label` för det de är, och prova varje ny vy med enbart tangentbordet innan den släpps.

Läs mer

- Architecture of the World Wide Web, Volume One — W3C-rekommendationen om user agents och om hur arbetet delas mellan klient och server.
- HTML Standard — WHATWG:s levande standard för de element och webbläsar-API:er som en frontend för webben byggs av.
- Web Content Accessibility Guidelines (WCAG) 2.2 — W3C-rekommendationen med de testbara framgångskriterierna bakom ett tillgängligt gränssnitt.
- Content Security Policy Level 3 — W3C:s arbetsutkast som definierar nonces och `strict-dynamic`.
- Next.js: Server and Client Components — hur ett ramverk delar upp frontendens kod mellan server och webbläsare.

Ordlista

Välj utifrån era faktiska förutsättningar

Beskriv arbetsflödet, befintliga gränssnitt och användarna som behöver förändringen. Vi kan definiera en testbar leverans och en överlämning som gör att teamet kan underhålla lösningen.

Se vad det kostar Hör av er →

Gör begreppen praktiska

Fråga hur ”Vad är en frontend?” gäller i er organisation. Ett konkret exempel från ert nuvarande system räcker för att börja samtalet.

Se våra tjänster

welcome@engineer.company +45 50 10 61 84 Köpenhamn, Danmark

Ta med dig den här sidan

Visitkort

Omsorgsfullt skapad med kärlek av D.

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

Upphovsrätt © 2025 – nu · Engineer ApS · Alla rättigheter förbehållna.

Utskrivet från https://engineer.company/sv/glossary/frontend/