Engineer

Frontend

systemdelen som håndterer presentasjon og interaksjon

Hva det er

En frontend er den delen av et programvaresystem som folk ser på og betjener: sidene, skjermbildene, nettskjemaene og knappene, sammen med koden som tegner dem og reagerer på det brukeren gjør. Ordet skrives også front end eller front-end, og på weben kalles det samme området ofte klientsiden.

På weben settes en frontend sammen av tre språk som en nettleser forstår. HTML gir siden struktur og mening, CSS bestemmer hvordan den ser ut, og JavaScript legger til oppførsel. Nettleseren er det W3C-dokumentet Architecture of the World Wide Web kaller en brukeragent: programvare som handler på vegne av en person, henter data fra servere og presenterer dem. Alt en frontend sender dit, havner på en enhet som tilhører noen andre.

Motparten er backend, som lagrer data, anvender forretningsreglene og svarer på forespørsler. De to snakker sammen over HTTP, som regel gjennom et API. Det er fristende å trekke grensen mellom dem der nettverket går, men dagens rammeverk visker den ut. I Next.js er sider og layouter Server Components som standard: de produserer det brukeren ser, og kjøres på en server. Et `"use client"` -direktiv øverst i en fil markerer grensen, og den filen, det den importerer, og komponentene den rendrer, havner i bunten som sendes til nettleseren. Det som gjør kode til frontend, er jobben den gjør, presentasjon og interaksjon. Hvor den kjøres, er en egen beslutning.

Bruk begrepet i en beslutning

Bruk «Hva er en frontend?» i en konkret beslutning. Vi kan knytte begrepet til systemet dere bygger eller kjøper. Grensesnitt som laster raskt, fungerer på mobilen og oppfører seg likt i alle nettlesere kundene dine måtte bruke.

Se tjenestene våre Ta kontakt →

Undersøk antakelsene bak ordet

Bruk forklaringen og det tilknyttede arbeidet til å formulere spørsmål for prosjektet. Vi kan undersøke antakelsene sammen før et verktøy velges. Styrte fullstack-utviklingen av GIS-kart — fullstack-kartutvikling, fra database til nettleser

Styrte fullstack-utviklingen av GIS-kart Ta kontakt →

Hvorfor det er viktig

Den første beslutningen er hvor arbeidet skjer. W3C-arkitekturdokumentet beskriver et spekter med en klient som gjør alt i den ene enden og en server som gjør alt i den andre, og konstaterer at de fleste løsninger kombinerer de to. Hvor en applikasjon hører hjemme, avhenger etter dokumentets ord av hvor kraftig klienten er, hvor kraftig og belastet serveren er, og båndbredden mellom dem. Avveiningen beskrives fra begge sider: innhold som skreddersys på serveren, gjør caching mindre effektiv, og når antallet mulige klienter er ubegrenset, skalerer det bedre å skyve beregningene ut til dem. De vanlige posisjonene er sider bygd én gang, sider rendret på en server per forespørsel, sider satt sammen i nettleseren, og blandinger av disse. En side som er lik for alle, er billigst å bygge én gang og legge i cache. En side som viser én kundes data, rendres per forespørsel et sted, og velger du nettleseren, må hver leser laste ned koden som gjør jobben.

Den andre beslutningen er hvor mye du kan stole på frontenden, og svaret som holder i praksis, er at serveren stoler på ingenting av den. Alt en frontend leverer, kan leses, endres og spilles av på nytt av den som sitter med nettleseren. En skjult knapp eller et skjema som nekter å sende, er en høflighet mot en ærlig bruker og en liten omvei for alle andre, så hver tillatelse og hver validering som betyr noe, gjentas på serveren. Hemmeligheter følger samme regel: en API-nøkkel som havner i bunten, er offentlig, uansett hva variabelen heter. Content Security Policy snevrer inn hva et injisert skript kan gjøre, og spesifikasjonen presenterer den som dybdeforsvar, aldri som erstatning for grundig validering av inndata og koding av utdata.

Frontenden er også der universell utforming vinnes eller tapes. WCAG 2.2, W3C-anbefalingen som ble publisert i desember 2024, setter opp testbare suksesskriterier, og mange av dem avgjøres av valg i frontenden: om en kontroll kan nås og betjenes fra tastaturet, og om den gir et navn og en rolle videre til hjelpemidler som en skjermleser. Innebygde HTML-elementer oppfyller flere av dem uten ekstra arbeid, mens en egenutviklet widget må bygge opp hvert av dem for hånd. For en norsk leser kommer loven i tillegg: ifølge Uutilsynet skal ikt-

løsninger i Norge være universelt utformet, nettsider og apper medregnet, og både private og offentlige virksomheter må følge regelverket, der kravene er hentet fra WCAG-standarden.

Til slutt: en frontend som vet hvordan databasen lagrer ting, må endres hver gang databasen endres, mens en frontend som holder seg til presentasjon, lar hver side bevege seg i sitt eget tempo.

Slik ser det ut i praksis

Hvor en frontend rendrer, kan avgjøres del for del. På én plattform velger Next.js 16-frontenden riktig renderingsmodus for hver del av appen. Markedsføringssider og offentlige sider genereres statisk, og de interaktive delene rendres i nettleseren. På det innloggede dashbordet og de store katalogoversiktene, der N+1-mønsteret gjorde én side om til en kaskade av rundturer, henter en fabrikk kalt `createPrefetchedServerPage` sidens data på serveren og gir dem til klienten ferdig utfylt, og en strategi for cache-invalidering er dokumentert for hver spørring. En side som er flyttet over på fabrikken, lastes med én rundtur, og den neste som flyttes, arver den oppførselen. Siden presenterer dette som en migrering underveis: resten av den innloggede appen er fortsatt klientrendret og venter på tur etter den.

Grunnleggerens arkitekturbidrag hos NextMariner, en maritim plattform, gir frontenden én oppgave og ingenting annet. Next.js-frontenden tar seg av presentasjon og interaktivitet. Et bevisst tynt Go-API, bygd med Huma over Fiber, ruter og validerer forespørsler, anvender sikkerhetsfiltrering og orkestrerer, mens forretningslogikken ligger i PostgreSQL-funksjoner. Utformingen legger en endring av en forretningsregel i ett lag, i SQL, i stedet for å kreve den samme regelen i alle tre lagene. Grensene ble skrevet ned og deretter håndhevet i kodegjennomganger. Det dokumenterte resultatet er at presentasjon, behandling av forespørsler og forretningsregler har atskilte ansvarsområder.

Herdingen av en applikasjon tar utgangspunkt i at klienten er fiendtlig. Det frontenden selv bidrar med, sitter i proxy-mellomvaren `proxy.ts` i Next.js: HSTS, SameSite-cookies og en Content Security Policy der en ny nonce lages for hver forespørsel og `strict-dynamic` er slått på. Selve grensen ligger på serveren. Rettighetene, som dekker abonnementsnivå, rolle, organisasjon og avgrensning per skip, kontrolleres på nytt på serveren ved hver forespørsel, og sperrene i frontenden teller bare som brukeropplevelse: de former hva en person ser, og serveren avgjør hva den personen får gjøre.

Hvor det går galt

Når en frontend svikter, er årsaken gjerne den samme: noe ble avgjort i nettleseren som hørte hjemme på serveren, eller omvendt.

En API-nøkkel dukker opp i JavaScript som enhver besøkende kan laste ned. I Next.js når en miljøvariabel nettleserbunten bare når navnet begynner med `NEXT_PUBLIC_`, og alle andre variabler som leses fra klientkode, erstattes med en tom streng. En forespørsel som kommer tilbake som uautorisert, fulgt av en rask navneendring for å få det til å virke, legger hemmeligheten rett inn i en offentlig fil. Hold kode som trenger en hemmelighet, på serveren, og merk modulen med pakken `server-only`, slik at en import fra en Client Component får bygget til å feile. Bytt ut enhver nøkkel som allerede har nådd nettleseren.

```
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 bruker når poster som grensesnittet hadde skjult for dem. Tilgangssjekken lå i frontenden, som en rutevakt, et skjult menypunkt eller en deaktivert knapp. Den som leser nettverkstrafikken, kan kalle API-et direkte med andre parametere. Sjekk hver rettighet på serveren ved hver forespørsel, mot den innloggede brukeren og den konkrete posten, og behandle frontendens sjekker som en bekvemmelighet for ærlige brukere.

Siden fylles ut én spinner om gangen. Hver komponent henter sine egne data etter at den er rendret, så barnets forespørsel venter på forelderens, og siden kommer fram som en kaskade av rundturer. Hent sidens data samlet og nær kilden, på serveren der rammeverket tillater det, og send dem nedover.

Tastatur- og skjermleserbrukere blir sittende fast i en egenutviklet kontroll. En klikkbar `div` får museoppførselen til en knapp og ingenting av resten: den tar ikke fokus, overser Enter og mellomromstasten, og forteller hjelpemidlene ingenting om hva den er. WCAG 2.2 krever på nivå A at all funksjonalitet virker fra

tastaturet (suksesskriterium 2.1.1), og at hver komponent gir fra seg navn og rolle (4.1.2). Bruk `button`, `a`, `input` og `label` til det de er laget for, og prøv hvert nytt skjermbilde med bare tastaturet før det går ut.

Les mer

- Architecture of the World Wide Web, Volume One — W3C-anbefalingen om brukeragenter og om hvordan arbeidet fordeles mellom klient og server.
- HTML Standard — WHATWGs levende standard for elementene og nettleser-API-ene en webfrontend bygges av.
- Web Content Accessibility Guidelines (WCAG) 2.2 — W3C-anbefalingen med de testbare suksesskriteriene bak et tilgjengelig grensesnitt.
- Content Security Policy Level 3 — W3C-arbeidsutkastet som definerer nonces og `strict-dynamic`.
- Next.js: Server and Client Components — hvordan ett rammeverk deler frontendens kode mellom server og nettleser.
- Regelverk og krav — Uutilsynets oversikt over kravene til universell utforming av ikt i Norge, med WCAG-kravene som gjelder.

Ordliste

Velg ut fra deres faktiske rammer

Beskriv arbeidsflyten, eksisterende grensesnitt og brukerne som trenger endringen. Vi kan definere en testbar leveranse og en overlevering som gjør teamet i stand til å vedlikeholde løsningen.

Se hva det koster Ta kontakt →

Gjør begrepene praktiske

Spør hvordan «Hva er en frontend?» gjelder i organisasjonen deres. Et konkret eksempel fra dagens system er nok til å starte samtalen.

Se tjenestene våre

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

Ta med deg denne siden

Visittkort

Laget med kjærlighet av D.

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

Opphavsrett © 2025 – nå · Engineer ApS · Alle rettigheter forbeholdt.

Skrevet ut fra https://engineer.company/nb/glossary/frontend/