**Engineer**

# Frontend

partie du logiciel chargée de l'interface utilisateur

## Ce que c’est

Le frontend est la partie d’un système logiciel que les personnes voient et utilisent : les pages, écrans, formulaires et commandes, ainsi que le code qui les dessine et réagit aux actions de l’utilisateur. Le terme s’écrit aussi front end ou front-end. Sur le web, ce périmètre est souvent appelé côté client.

Sur le web, un frontend est assemblé à partir de trois langages compris par le navigateur. HTML donne à la page sa structure et son sens, CSS définit son apparence et JavaScript ajoute les comportements. Le navigateur est ce que l’Architecture du World Wide Web du W3C appelle un agent utilisateur : un logiciel agissant pour le compte d’une personne, qui récupère des données sur des serveurs et les présente. Tout ce qu’un frontend lui transmet arrive sur un appareil appartenant à quelqu’un d’autre.

Son partenaire est le backend, qui stocke les données, applique les règles métier et répond aux requêtes. Les deux communiquent par HTTP, généralement à travers une API. Il est tentant de placer leur frontière au niveau du réseau, mais les frameworks actuels la brouillent. Dans Next.js, les pages et les modèles de mise en page sont, sans indication contraire, des Server Components : ils produisent ce que voit l’utilisateur et s’exécutent sur un serveur. Une directive `"use client"` en haut d’un fichier marque la frontière : ce fichier, ses imports et les composants qu’il rend rejoignent le bundle envoyé au navigateur. Ce qui rattache ce code au frontend, c’est sa fonction de présentation et d’interaction ; son lieu d’exécution relève d’un choix distinct.

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce qu'un frontend ? » pour éclairer une décision réelle. Nous pouvons vous aider à relier ce concept au système que vous construisez ou achetez. Des interfaces rapides, utilisables sur téléphone et cohérentes dans tous les navigateurs de vos clients.

**Découvrir nos prestations** Contactez-nous →

Vérifiez les hypothèses derrière le terme

Appuyez-vous sur l'explication et les réalisations liées pour formuler les questions de votre projet. Nous pouvons examiner les hypothèses avec vous avant le choix d'un outil. Développement cartographique GIS full-stack — cartographie full-stack, de la base au navigateur

**Développement cartographique GIS full-stack** Contactez-nous →

# Pourquoi c’est important

Le premier choix porte sur le lieu où le travail s’effectue. Le document architectural du W3C décrit un éventail allant du client qui fait tout au serveur qui fait tout, et constate que la plupart des conceptions combinent les deux. La place d’une application dans cet éventail dépend, selon le document, de la puissance du client, de la puissance et de la charge du serveur, ainsi que de la bande passante qui les relie. Il expose les deux faces du compromis : adapter le contenu sur le serveur réduit l’efficacité du cache ; lorsque le nombre de clients possibles est sans limite, leur déléguer les calculs facilite le passage à l’échelle. Les positions habituelles sont les pages construites une fois, les pages rendues sur le serveur à chaque requête, les pages assemblées dans le navigateur et leurs combinaisons. Une page identique pour tous coûte moins cher si elle est construite une fois et mise en cache. Une page présentant les données d’un client est rendue quelque part à chaque requête ; choisir le navigateur signifie que chaque lecteur télécharge le code chargé de ce rendu.

Le deuxième choix concerne la confiance qu’on lui accorde. En pratique, le serveur ne lui fait aucune confiance. Tout ce qu’un frontend livre peut être lu, modifié et rejoué par la personne qui contrôle le navigateur. Un bouton caché ou un formulaire qui refuse l’envoi facilite l’usage honnête et ne représente qu’un petit détour pour les autres. Tous les contrôles d’autorisation et de validation importants sont donc répétés sur le serveur. La même règle vaut pour les secrets : une clé d’API qui arrive dans le bundle est publique, quel que soit le nom de la variable. Content Security Policy limite ce qu’un script injecté peut faire. Sa spécification la présente comme une défense en profondeur, jamais comme un substitut à une validation des entrées et à un encodage des sorties soigneux.

Le frontend est aussi le lieu où l’accessibilité se gagne ou se perd. WCAG 2.2, la recommandation du W3C publiée en décembre 2024, définit des critères de succès testables. Beaucoup dépendent de choix du frontend : peut-on atteindre et utiliser une commande au clavier, expose-t-elle un nom et un rôle aux technologies d’assistance ? Les éléments HTML natifs en satisfont plusieurs d’emblée ; un composant personnalisé doit les reconstruire un par un, à la main.

Enfin, un frontend qui connaît la façon dont la base stocke ses données doit changer chaque fois que cette organisation change. S'il s'en tient à la présentation, chaque côté peut évoluer à son rythme.

## À quoi cela ressemble en pratique

Sur une plateforme, le lieu du rendu est décidé partie par partie. Son frontend Next.js 16 choisit un mode de rendu pour chaque partie de l'application. Les pages marketing et publiques sont générées statiquement ; les parties interactives sont rendues dans le navigateur. Sur le tableau de bord accessible après authentification et les longues listes d'annuaire, le schéma N+1 transformait une page en cascade d'allers-retours. Une fabrique nommée `createPrefetchedServerPage` récupère les données de la page sur le serveur et les transmet déjà renseignées au client. Une stratégie d'invalidation du cache est documentée pour chaque requête de données. Une page passée sur cette fabrique se charge en un aller-retour ; la suivante hérite de ce comportement. La migration se poursuit : le reste de l'application authentifiée est encore rendu côté client et attend son tour.

La contribution architecturale de la fondatrice à NextMariner, une plateforme maritime, ne donne qu'une seule fonction au frontend. Le frontend Next.js s'occupe de la présentation et de l'interactivité. Une API Go aux responsabilités volontairement limitées, construite avec Huma sur Fiber, achemine et valide les requêtes, applique le filtrage de sécurité et assure l'orchestration, tandis que la logique métier réside dans des fonctions PostgreSQL. Un changement de règle métier se fait ainsi dans une seule couche, en SQL, au lieu d'exiger la même règle dans les trois. Les frontières ont été consignées, puis imposées lors des revues de code. Le résultat documenté est une séparation des responsabilités entre présentation, traitement des requêtes et règles métier.

Le travail de sécurisation d'une application part du principe que le client est hostile. Pour le frontend, le middleware proxy Next.js dans `proxy.ts` définit une Content Security Policy avec un nonce généré à chaque requête et `strict-dynamic`, ainsi que HSTS et des cookies SameSite. La frontière de sécurité elle-même se situe sur le serveur. Les droits, définis selon le niveau d'offre, le rôle, l'organisation et le périmètre des navires, y sont revérifiés à chaque requête. Les restrictions du frontend ne relèvent que de l'expérience utilisateur : elles façonnent ce qu'une personne voit ; le serveur décide de ce qu'elle peut faire.

# Où les problèmes surviennent

Les défaillances du frontend ont souvent la même cause : une décision a été prise dans le navigateur alors qu'elle relevait du serveur, ou l'inverse.

**Une clé d'API se retrouve dans le JavaScript téléchargeable par n'importe quel visiteur.** Dans Next.js, une variable d'environnement n'arrive dans le bundle du navigateur que si son nom commence par `NEXT_PUBLIC_`. Toute autre variable lue par le code client est remplacée par une chaîne vide. Une requête échoue faute d'autorisation ; renommer vite la variable pour la faire fonctionner incorpore le secret dans un fichier public. Gardez sur le serveur le code qui a besoin d'un secret, et marquez son module avec le paquet `server-only` : son import depuis un Client Component fera alors échouer la compilation. Renouvelez toute clé déjà livrée.

```
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();
}
```

**Un utilisateur atteint des enregistrements que l'interface lui cachait.** Le contrôle d'autorisation se trouvait dans le frontend : protection de route, entrée de menu cachée ou bouton désactivé. Quiconque observe le trafic réseau peut appeler directement l'API avec d'autres paramètres. Vérifiez chaque droit sur le serveur à chaque requête, pour l'utilisateur connecté et l'enregistrement visé. Considérez les contrôles du frontend comme une aide pour les utilisateurs honnêtes.

**La page se remplit un indicateur de chargement après l'autre.** Chaque composant récupère ses propres données après son rendu : la requête d'un enfant attend celle de son parent, et la page arrive en cascade d'allers-retours. Récupérez les données d'une page ensemble, au plus près de leur source, sur le serveur lorsque le framework le permet, puis transmettez-les aux composants.

**Les personnes utilisant le clavier ou un lecteur d'écran restent bloquées sur une commande personnalisée.** Un `div` cliquable acquiert le comportement d'un bouton à la souris, sans le reste : il ne reçoit pas le focus, ignore Entrée et

Espace, et n'indique pas sa nature aux technologies d'assistance. Au niveau A, WCAG 2.2 exige que toutes les fonctionnalités soient utilisables au clavier (critère de succès 2.1.1) et que chaque composant expose son nom et son rôle (4.1.2). Utilisez `button`, `a`, `input` et `label` selon leur fonction, et parcourez chaque nouvel écran au seul clavier avant sa mise en service.

## Pour aller plus loin

- Architecture of the World Wide Web, Volume One — la recommandation du W3C sur les agents utilisateurs et le partage du travail entre client et serveur.
- HTML Standard — le standard vivant du WHATWG pour les éléments et les API du navigateur qui constituent un frontend web.
- Web Content Accessibility Guidelines (WCAG) 2.2 — la recommandation du W3C et ses critères de succès testables pour une interface accessible.
- Content Security Policy Level 3 — le projet de spécification du W3C qui définit les nonces et `strict-dynamic`.
- Next.js : Server et Client Components — la façon dont un framework répartit le code d'un frontend entre serveur et navigateur.

Glossaire

Choisissez selon vos contraintes réelles

Présentez le processus de travail, les interfaces existantes et les utilisateurs qui ont besoin du changement. Nous pouvons définir une livraison testable et une transmission permettant à votre équipe d'assurer la maintenance des éléments livrés.

**Consulter les tarifs** Contactez-nous →

# Rendez la terminologie concrète

Demandez comment « Qu'est-ce qu'un frontend ? » s'applique à votre organisation. Un exemple concret tiré de votre système actuel suffit pour engager la discussion.

**Découvrir nos prestations**

welcome@engineer.company +45 50 10 61 84 Copenhague, Danemark

Emportez cette page avec vous

Fiche de contact

Créé avec soin et amour par D.

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

Droit d'auteur © 2025 – aujourd'hui · Engineer ApS · Tous droits réservés.

Imprimé depuis https://engineer.company/fr/glossary/frontend/