**Engineer**

# Next.js

N Framework React pour construire des applications web

## Ce que c’est

Next.js est un framework qui enveloppe React, la bibliothèque qui permet de construire des interfaces à partir de composants, et fournit ce que React laisse à chaque application : routage, rendu serveur, récupération des données et construction préconfigurée. Le paquet et sa commande s’appellent tous deux `next`, comme dans `next dev` et `next build`.

Sa première version est sortie en 2016 pour, selon ses auteurs, « offrir une façon simple de rendre les applications React côté serveur ». La conception initiale reposait sur quelques principes : aucune configuration initiale, système de fichiers comme API — un fichier dans `pages/` devenait une route —, rendu serveur et découpage du code automatiques, récupération des données laissée au développeur. En 2023, les notes de version créditaient plus de 2 600 développeurs aux côtés d’une équipe centrale chez Vercel.

Next.js 13, le 25 octobre 2022, a ajouté en bêta un répertoire `app` fondé sur les React Server Components, avec des mises en page imbriquées et une diffusion en continu. Next.js 13.4, le 4 mai 2023, a déclaré cet App Router stable ; la documentation décrit depuis deux routeurs. Le Pages Router est « le routeur d’origine, toujours pris en charge et en cours d’amélioration ». Même leurs versions de React diffèrent : l’App Router intègre des versions canary de React, tandis que le Pages Router utilise celle indiquée dans le `package.json` du projet. Dans l’App Router, pages et mises en page sont rendues côté serveur, sauf si un fichier choisit le navigateur avec la directive `"use client"`.

Next.js 16 est arrivé le 21 octobre 2025. Il a fait de Turbopack l’outil de regroupement par défaut, renommé `middleware.ts` en `proxy.ts` et introduit Cache Components : une directive `"use cache"` marque ce qui peut être mis en cache, tout le reste étant rendu au moment de la requête. Next.js 16.4, le 6

octobre 2026, recommande Cache Components pour chaque application, l'active pour les nouveaux projets de `create-next-app` et annonce qu'il « deviendra le choix par défaut dans Next.js 17 ».

Utilisez le concept pour prendre une décision

Utilisez « Qu'est-ce que Next.js ? » 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 choix auquel répond Next.js est de confier à un framework le routage, le rendu et le serveur d'une application React, ou d'assembler ces éléments vous-même en gardant la maîtrise des choix.

Il propose des conventions à la place de la configuration. Les routes viennent des fichiers, outils de regroupement et compilateurs sont configurés automatiquement, et chaque route peut être construite à l'avance, rendue côté serveur à chaque requête ou dans le navigateur. Les Server Components gardent l'accès aux données et les secrets sur le serveur et envoient moins de JavaScript au navigateur. Pour une application mêlant pages publiques, espace connecté et backend propre, ce choix par route est l'attrait principal.

Le premier coût est un serveur, avec Node.js 20.9 ou ultérieur pour Next.js 16. Le réglage `output: 'export'` écrit un fichier HTML par route pour n'importe quel serveur web statique ; le guide le présente comme un moyen de commencer en statique puis d'ajouter des fonctions serveur. Mais la liste des fonctionnalités qui nécessitent un serveur Node.js couvre une grande part de ce qui distingue Next.js : Proxy, Server Actions, cookies, réécritures, redirections, en-têtes, Incremental Static Regeneration et optimisation des images avec le chargeur par défaut.

Le deuxième coût est le rythme. Quatre versions apportant de nouvelles fonctionnalités ont suivi la 16.0 en un an et le modèle de cache va encore changer de fonctionnement par défaut. Les notes de la 16.4 disent que Cache Components remplace « les comportements de cache implicites des versions

précédentes de l'App Router » : le code écrit pour Next.js 13 à 15 suit donc d'autres règles de cache. Les tutoriels en ligne sont aussi partagés entre les deux routeurs.

Le troisième coût est que les mises à niveau relèvent de la sécurité. Le 3 décembre 2025, le projet a publié un avis sur CVE-2025-66478, vulnérabilité critique du protocole React Server Components, demandant de passer immédiatement aux versions corrigées ; le blog a publié trois autres billets de sécurité entre le 30 septembre et le 14 octobre 2026. Une équipe adoptant Next.js s'engage à suivre ses versions de près. Une équipe publiant des documents identiques pour tous les visiteurs paie un modèle serveur qu'elle utilise à peine.

## À quoi cela ressemble en pratique

Dans le travail de la fondatrice chez NextMariner, le frontend Next.js 16 rend chaque surface à sa façon : le choix par route est mis en pratique. Les pages marketing et les pages publiques sont générées statiquement, les parties réellement interactives restent rendues côté client et, pour le tableau de bord authentifié et les grandes listes d'annuaire, une fabrique, `createPrefetchedServerPage`, récupère les données côté serveur et les transmet déjà chargées au client. Les pages passent à la fabrique une à une ; la page précise que le reste de l'application authentifiée est encore rendu côté client et attend son tour : « une migration avec un mécanisme opérationnel et une prochaine page évidente, plutôt qu'une conversion entièrement achevée ». L'App Router a été conçu pour ce type de migration : les notes de la 13.4 décrivent son adoption progressive.

Une suite de tests par couches donne au frontend Next.js sa propre couche : 543 tests Vitest et Playwright, dont 48 fichiers de bout en bout et 30 tests navigateur, à côté de couches distinctes pour l'API Go et la base de données. Chaque couche est vérifiée là où son comportement s'exécute, plutôt que de faire passer tous les tests par le navigateur : cela compte dans un framework dont l'arbre de composants tourne en partie sur le serveur et en partie dans le navigateur.

Quand la fondatrice, travaillant chez NextMariner, a renforcé la sécurité d'une application, les protections destinées au navigateur ont été placées dans `proxy.ts`, le middleware renommé. Il définit une politique Content-Security-Policy avec un nonce par requête et `strict-dynamic`, ainsi que HSTS et des

cookies SameSite. L'autorisation se trouve ailleurs : les droits, couvrant le niveau, le rôle, l'organisation et le périmètre du navire, sont revérifiés côté serveur à chaque requête ; les contrôles du frontend ne comptent que pour l'expérience utilisateur. Cette séparation suit la référence Proxy, qui présente la fonction comme un dernier recours et demande de vérifier l'autorisation « dans chaque Server Function plutôt que de s'appuyer uniquement sur Proxy ».

## Où les problèmes surviennent

**Après l'ajout d'une redirection de connexion, la page de connexion s'affiche sans ses styles.** Sans `matcher`, Proxy s'exécute à chaque requête, y compris les fichiers construits sous `_next/static`, les images optimisées sous `_next/image` et tout `public/` : une redirection destinée aux pages attrape aussi le CSS, le JavaScript et les images nécessaires à la connexion. Donnez à Proxy un matcher excluant ces chemins :

```
// proxy.ts
export const config = {
    matcher: ["/((?!_next/static|_next/image|favicon.ico).*)"],
};
```

Les fichiers de `public/` ne partagent aucun préfixe : ajoutez donc aussi leurs chemins.

**Une action de formulaire fonctionne pour un utilisateur déconnecté sur une route apparemment protégée.** Les Server Functions n'ont pas de routes propres : elles arrivent comme requêtes POST à la page qui les utilise. Un matcher excluant un chemin exclut donc aussi les fonctions qui s'y trouvent. La référence Proxy avertit que changer le matcher ou déplacer une fonction vers une autre route pendant une refactorisation « peut retirer silencieusement la couverture de Proxy ». Vérifiez session et permissions dans chaque Server Function et traitez Proxy comme une couche extérieure, jamais comme l'unique couche.

**Un utilisateur enregistre un changement et la page affiche encore l'ancienne valeur.** Dans Next.js 16, `revalidateTag(tag, 'max')` donne aux entrées étiquetées un comportement stale-while-revalidate : la copie en cache reste servie pendant la récupération d'une nouvelle, ce qui convient au contenu tolérant une cohérence à terme et surprend la personne qui vient de cliquer sur « Enregistrer ». Appelez `updateTag(tag)` depuis la Server Action ayant modifié les

données : l'entrée expire et les données fraîches sont lues dans la même requête. L'ancien `revalidateTag(tag)` à un argument est déconseillé et produit désormais une erreur TypeScript.

**Le codemod de mise à niveau se termine, mais la construction échoue encore sur `params` ou `cookies()`.** Next.js 15 a rendu les API de requête asynchrones avec une compatibilité synchrone temporaire ; Next.js 16 la retire. Le codemod `upgrade` ne lance pas toutes les migrations : exécutez aussi `npx @next/codemod@canary next-async-request-api .`, puis `npx next typegen` pour générer les auxiliaires `PageProps`, `LayoutProps` et `RouteContext`. La même mise à niveau supprime `next lint` et retire l'analyse statique de `next build` : ajoutez donc ESLint ou Biome comme étape distincte.

## Pour aller plus loin

- Documentation Next.js — documentation officielle avec bascule entre les versions App Router et Pages Router de chaque page.
- Next.js 16 — version apportant Cache Components, `proxy.ts` et Turbopack comme outil de regroupement par défaut, avec les ruptures de compatibilité.
- Passer à la version 16 — codemods, versions requises et chaque API retirée ou renommée.
- proxy.js — matchers, place de Proxy dans l'ordre des requêtes et raison du renommage du middleware.
- Next.js 16.4 — arguments pour Cache Components et projet d'en faire le choix par défaut dans Next.js 17.
- Créer un export statique — fonctions nécessitant un serveur et ce qu'écrit `output: 'export'`.

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 que Next.js ? » 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/nextjs/