

Engineer

Next.js

N React-Framework zum Entwickeln von Webanwendungen



Was es ist

Next.js ist ein Framework auf Grundlage von React, der Bibliothek zum Aufbau von Benutzeroberflächen aus Komponenten. Es liefert die Teile, die React jeder Anwendung selbst überlässt: Routing, serverseitiges Rendern, Datenabruf und einen fertig konfigurierten Build. Das Paket und sein Befehl heißen beide `next`, wie in `next dev` und `next build`.

Es erschien erstmals 2016, um in den Worten seiner Autoren „einen einfachen Weg zum serverseitigen Rendern von React-Anwendungen zu bieten“. Der frühe Entwurf beruhte auf wenigen Prinzipien: keine Einrichtung, das Dateisystem als API, sodass eine Datei im Ordner `pages/` zur Route wurde, automatisches serverseitiges Rendern und Code-Aufteilung sowie Datenabruf in der Verantwortung der Entwickler. 2023 nannten die Versionshinweise neben einem Kernteam bei Vercel bereits über 2.600 Mitwirkende.

Next.js 13 ergänzte am 25. Oktober 2022 ein `app`-Verzeichnis als Beta, aufgebaut auf React Server Components, mit verschachtelten Layouts und Streaming. Next.js 13.4 erklärte diesen App Router am 4. Mai 2023 für stabil. Seitdem beschreibt die Dokumentation zwei Router. Der Pages Router ist „der ursprüngliche Router, weiterhin unterstützt und verbessert“. Sie unterscheiden sich sogar in der verwendeten React-Version: Der App Router liefert integrierte React-Canary-Versionen mit, der Pages Router nutzt die in der `package.json` des Projekts genannte Version. Im App Router werden Seiten und Layouts auf dem Server gerendert, sofern eine Datei nicht mit der Direktive `"use client"` ausdrücklich den Browser wählt.

Next.js 16 erschien am 21. Oktober 2025. Es machte Turbopack zum Standard-Bundler, benannte `middleware.ts` in `proxy.ts` um und führte Cache Components ein. In diesem Modell markiert die Direktive `"use cache"`, was

zwischengespeichert werden darf; alles andere wird zur Anfragezeit gerendert. Next.js 16.4 vom 6. Oktober 2026 empfiehlt Cache Components für jede Anwendung, aktiviert sie für neue, mit `create-next-app` erzeugte Projekte und sagt, sie würden „in Next.js 17 zum Standard“.

Das Konzept für eine Entscheidung nutzen

Nutzen Sie „Was ist Next.js?“ für eine konkrete Entscheidung. Wir können helfen, das Konzept mit dem System in Verbindung zu bringen, das Sie entwickeln oder einkaufen. Oberflächen, die schnell laden, auf dem Smartphone funktionieren und sich in allen Browsern Ihrer Kunden gleich verhalten.

Unsere Leistungen entdecken Kontakt aufnehmen →

Die Annahmen hinter dem Begriff prüfen

Nutzen Sie die Erklärung und die verlinkten Arbeiten, um Fragen für Ihr Projekt zu formulieren. Wir können die Annahmen gemeinsam mit Ihnen prüfen, bevor ein Werkzeug ausgewählt wird. Full-Stack-GIS-Kartenentwicklung geleitet — Full-Stack-Kartenentwicklung von der Datenbank bis zum Browser

Full-Stack-GIS-Kartenentwicklung geleitet Kontakt aufnehmen →

Warum es wichtig ist

Next.js beantwortet die Frage, ob ein Framework Routing, Rendering und Server einer React-Anwendung übernehmen soll oder ob man diese Teile selbst zusammensetzt und die Entscheidungen behält.

Es bietet Konventionen statt Konfiguration. Routen entstehen aus Dateien, Bundler und Compiler werden automatisch eingerichtet, und jede Route lässt sich vorab erzeugen, für jede Anfrage serverseitig oder im Browser rendern. Server Components halten Datenzugriff und Geheimnisse auf dem Server und senden weniger JavaScript an den Browser. Für eine Anwendung mit öffentlichen Seiten, einem Bereich für angemeldete Nutzer und eigenem Backend ist diese Entscheidung pro Route der wesentliche Reiz.

Der erste Preis ist ein Server mit Node.js 20.9 oder neuer für Next.js 16. Die Einstellung `output: 'export'` schreibt pro Route eine HTML-Datei für beliebige statische Webserver. Der Leitfaden bewirbt das als Weg, statisch anzufangen und Serverfunktionen später hinzuzufügen. Die Liste der Funktionen, die einen Node.js-Server brauchen, umfasst allerdings vieles, was Next.js ausmacht: Proxy, Server Actions, Cookies, Umschreibungen, Weiterleitungen, Header, Incremental Static Regeneration und Bildoptimierung mit dem Standard-Loader.

Der zweite Preis ist das Tempo. Innerhalb eines Jahres folgten vier Funktionsversionen auf 16.0, und das Cache-Modell steht schon wieder vor einem Wechsel der Standardeinstellung. Die Hinweise zu 16.4 sagen, Cache

Components ersetzten „das implizite Cache-Verhalten der vorherigen App-Router-Versionen“. Für Next.js 13 bis 15 geschriebener Code speichert deshalb nach anderen Regeln zwischen. Auch die Anleitungen im Netz verteilen sich auf die beiden Router.

Der dritte Preis ist, dass Upgrades Sicherheitsarbeit sind. Am 3. Dezember 2025 veröffentlichte das Projekt einen Sicherheitshinweis zu CVE-2025-66478, einer kritischen Schwachstelle im React-Server-Components-Protokoll, und forderte zum sofortigen Upgrade auf korrigierte Versionen auf. Zwischen dem 30. September und dem 14. Oktober 2026 erschienen im Blog drei weitere Sicherheitsbeiträge. Ein Team, das Next.js übernimmt, verpflichtet sich damit, die Veröffentlichungen aufmerksam zu verfolgen. Ein Team, das Dokumente mit demselben Inhalt für alle Besucher veröffentlicht, bezahlt für ein Servermodell, das es kaum nutzt.

Wie es in der Praxis aussieht

Im Rahmen der Arbeit der Gründerin bei NextMariner rendert das Next.js-16-Frontend jede Oberfläche auf ihre eigene Weise. So wird die Wahl pro Route praktisch genutzt. Marketing- und öffentliche Seiten werden statisch erzeugt, die tatsächlich interaktiven Teile bleiben clientseitig gerendert. Beim Dashboard für angemeldete Nutzer und den großen Verzeichnislisten ruft eine Factory, `createPrefetchedServerPage`, die Seitendaten serverseitig ab und übergibt sie dem Client, der seine Daten damit bereits hat. Seiten wechseln einzeln auf die Factory. Die Seite sagt offen, dass der Rest des anmeldepflichtigen Anwendungsbereichs weiterhin clientseitig gerendert wird und dahinter auf die Umstellung wartet: „eine Migration mit funktionierendem Mechanismus und einer klar erkennbaren nächsten Seite für die Umstellung statt einer abgeschlossenen Gesamtumstellung“. Der App Router wurde für solche Wechsel entworfen. Die Hinweise zu 13.4 beschreiben seine schrittweise Einführung.

Eine nach Schichten gegliederte Testsuite gibt dem Next.js-Frontend eine eigene Schicht: 543 Testspezifikationen für Vitest und Playwright, darunter 48 Ende-zu-Ende-Dateien und 30 Browser-Testspezifikationen, neben getrennten Schichten für die Go-API und die Datenbank. Jede Schicht wird dort geprüft, wo ihr Verhalten liegt, statt alles durch einen Browser zu prüfen. Das ist in einem Framework wichtig, dessen Komponentenbaum teils auf dem Server und teils im Browser läuft.

Als die bei NextMariner tätige Gründerin eine Anwendung absicherte, kamen die browserseitigen Schutzmaßnahmen in `proxy.ts`, die umbenannte Middleware. Sie setzt eine Content-Security-Policy mit einem Nonce pro Anfrage und `strict-dynamic` sowie HSTS und SameSite-Cookies. Die Autorisierung liegt anderswo: Berechtigungen nach Stufe, Rolle, Organisation und Schiff werden serverseitig bei jeder Anfrage erneut geprüft. Die Schranken im Frontend dienen allein der Benutzerführung (UX). Diese Trennung entspricht der Proxy-Referenz. Sie bezeichnet die Funktion als letztes Mittel und verlangt die Prüfung der Autorisierung „in jeder Server Function, statt sich allein auf Proxy zu verlassen“.

Wo es schiefgeht

Nach dem Einbau einer Anmeldeweiterleitung lädt die Login-Seite ohne Formatierung. Ohne `matcher` läuft Proxy bei jeder Anfrage, auch für Build-Dateien unter `_next/static`, optimierte Bilder unter `_next/image` und alles in `public/`. Eine für Seiten gedachte Weiterleitung erfasst damit auch CSS, JavaScript und Bilder, die die Anmeldeseite benötigt. Geben Sie Proxy einen Matcher, der diese Pfade ausnimmt:

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

Dateien aus `public/` haben kein gemeinsames Präfix. Ergänzen Sie deshalb auch ihre Pfade.

Eine Formularaktion funktioniert für abgemeldete Nutzer auf einer scheinbar geschützten Route. Server Functions haben keine eigenen Routen. Sie kommen als POST-Anfragen an die Seite, auf der sie verwendet werden. Ein Matcher, der einen Pfad ausnimmt, nimmt deshalb auch die Funktionen dort aus. Die Proxy-Referenz warnt, eine Änderung am Matcher oder das Verschieben einer Funktion auf eine andere Route beim Refactoring könne „die Proxy-Abdeckung unbemerkt entfernen“. Prüfen Sie Sitzung und Berechtigungen in jeder Server Function und behandeln Sie Proxy als äußere Schicht, nie als einzige.

Ein Nutzer speichert eine Änderung, und die Seite zeigt weiter den alten Wert. In Next.js 16 gibt `revalidateTag(tag, 'max')` markierten Einträgen Stale-While-Revalidate-Verhalten: Die Cache-Kopie wird weiter ausgeliefert, während

eine frische abgerufen wird. Das passt zu Inhalten, die erst mit Verzögerung erreichte Konsistenz vertragen, überrascht aber jemanden, der gerade Save, also Speichern, gedrückt hat. Rufen Sie `updateTag(tag)` aus der Server Action auf, die die Änderung vorgenommen hat. Der Aufruf lässt den Eintrag ablaufen und liest innerhalb derselben Anfrage frische Daten. Das alte `revalidateTag(tag)` mit einem Argument ist veraltet und inzwischen ein TypeScript-Fehler.

Der Upgrade-Codemod ist fertig, und der Build scheitert weiter an `params` oder `cookies()`. Next.js 15 machte die Anfrage-APIs asynchron und bot vorübergehend synchrone Kompatibilität. Next.js 16 entfernt diese. Der Codemod `upgrade` führt nicht jede Migration aus. Führen Sie deshalb zusätzlich `npx @next/codemod@canary next-async-request-api .` aus, danach `npx next typegen`, um die Hilfstypen `PageProps`, `LayoutProps` und `RouteContext` zu erzeugen. Dasselbe Upgrade entfernt `next lint` und lässt `next build` keine Lint-Prüfungen mehr ausführen. Ergänzen Sie ESLint oder Biome deshalb als eigenen Schritt.

Weiterführende Lektüre

- Next.js-Dokumentation — die offizielle Dokumentation, mit Umschalter zwischen App-Router- und Pages-Router-Fassung jeder Seite.
- Next.js 16 — die Version mit Cache Components, `proxy.ts` und Turbopack als Standard-Bundler samt inkompatiblen Änderungen.
- Auf Version 16 aktualisieren — die Codemods, Versionsanforderungen und alle entfernten oder umbenannten APIs.
- proxy.js — Matcher, die Stelle von Proxy in der Anfragereihenfolge und der Grund für die Umbenennung der Middleware.
- Next.js 16.4 — die Argumente für Cache Components und der Plan, sie in Next.js 17 zum Standard zu machen.
- Einen statischen Export erzeugen — welche Funktionen einen Server brauchen und was `output: 'export'` schreibt.

Glossar

Für Ihre tatsächlichen Rahmenbedingungen auswählen

Beschreiben Sie den Arbeitsablauf, die vorhandenen Schnittstellen und die Nutzer, die die Veränderung benötigen. Wir können eine prüfbare Lieferung definieren und die Übergabe so gestalten, dass Ihr Team die gelieferten Bestandteile warten kann.

Preise ansehen Kontakt aufnehmen →

Die Begriffe praktisch nutzbar machen

Fragen Sie, wie „Was ist Next.js?“ auf Ihre Organisation zutrifft. Ein konkretes Beispiel aus Ihrem bestehenden System reicht aus, um das Gespräch zu beginnen.

Unsere Leistungen entdecken

welcome@engineer.company +45 50 10 61 84 Kopenhagen, Dänemark

Nehmen Sie diese Seite mit

Kontaktkarte

Mit Sorgfalt und Liebe gestaltet von D.

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

Copyright © 2025 – heute · Engineer ApS · Alle Rechte vorbehalten.

Gedruckt von https://engineer.company/de/glossary/nextjs/