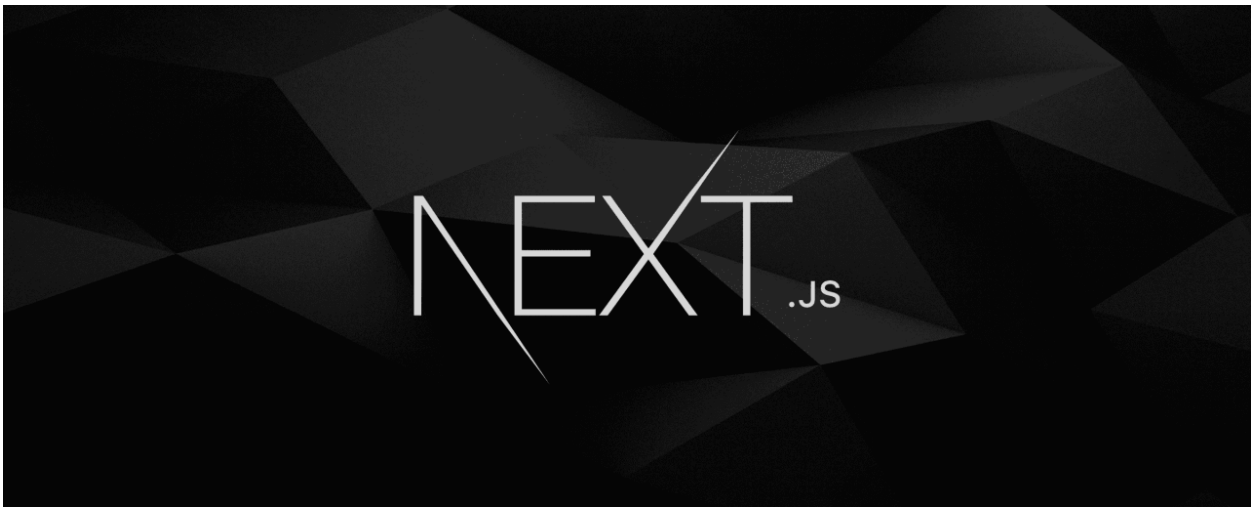

Next.js ?

by ZAKI MUSTHAFABILLAH | write with ‘Nation’

author : @ZAKI MUSHTHAFABILLAH

Blogs : https://www.astbyte.com/blog

🔍 Apa Sebenarnya Next.js Itu?

Jadi, Next.js itu bisa dibilang seperti upgrade-nya React. Kalau React ngasih kita tools buat bikin UI, Next.js ngasih kita satu paket lengkap buat bikin website yang cepet, SEO-friendly, dan bisa jalan di server maupun client. Cocok banget buat lo yang pengen bikin website dari blog sampai dashboard admin tanpa ribet setup backend.

💡 Kenapa Banyak yang Suka Next.js?

Gue pribadi suka Next.js karena:

- Kita gak perlu mikirin konfigurasi ribet-ribet, tinggal gas!
- Routing tinggal bikin file, gak usah setup react-router manual.
- Bisa bikin API langsung di project (jadi backend dan frontend bisa nyatu).
- Dan yang paling penting, performanya ngebut dan SEO-nya mantap.

🧩 Pages vs App Router (Mana yang Harus Dipakai?)

Pages Router

Klasik, simple, dan udah dipake banyak tutorial. Cocok buat lo yang baru mulai.

```
pages/
├── index.js      # /
├── about.js      # /about
├── blog/[slug].js # /blog/slug-dinamis
```

App Router

Mulai dari Next.js 13 ke atas, ada sistem routing baru yang lebih modular. Bisa pake layout per bagian dan lebih enak buat app besar.

```
app/
├── layout.js
├── page.js
└── blog/
    ├── page.js
    └── [slug]/
        └── page.js
```

Kalau gue pribadi? Mulai belajar dulu dari Pages Router, baru lanjut ke App Router kalau udah paham konsep SSR, layouting, dan modularisasi.

🔄 Cara Ambil Data (Data Fetching)

Next.js itu fleksibel banget, kita bisa ambil data langsung di server saat request masuk (SSR), atau di-build time (SSG). Ini tergantung kebutuhan:

- Kalau datanya sering berubah: **SSR**

- Kalau jarang berubah (misal halaman "Tentang Kami"): **SSG**
- Mau hybrid? Pakai ISR (regenerasi otomatis setelah sekian detik)

Contoh SSR

```
export async function getServerSideProps() {
  const res = await fetch('https://api.example.com');
  const data = await res.json();
  return { props: { data } };
}
```

Contoh SSG

```
export async function getStaticProps() {
  const res = await fetch('https://api.example.com/posts');
  const posts = await res.json();
  return { props: { posts }, revalidate: 60 };
}
```

⚙️ Buat API Sendiri Tanpa Backend Terpisah

Next.js bisa bikin API langsung di dalam folder `pages/api/`. Misalnya lo mau bikin route `GET /api/users`, cukup buat file:

```
// pages/api/users.js
export default function handler(req, res) {
  res.status(200).json([
    { id: 1, name: 'Zaki' },
    { id: 2, name: 'Mikumiestu' }
```

```
  ]);
}
```

Gak perlu install Express, gak perlu hosting backend terpisah. Simple banget buat proyek kecil sampai menengah.

🚀 Deployment Super Gampang

Next.js + Vercel = match made in heaven 💘

Langkah-langkah:

1. Push ke GitHub
2. Login ke vercel.com
3. Import project
4. Klik deploy dan... selesai.

Vercel bakal otomatis ngerti struktur Next.js lo dan siapin semuanya.

🧠 Tips Performance & SEO yang Sering Diremehin

- Jangan lupa pake `<Head>` dari `next/head` buat title dan meta tag.
- Gambar? Pakai `<Image />` dari `next/image`, otomatis optimize.
- Gunakan `Link` dari `next/link` untuk prefetch otomatis halaman.
- Tambahin revalidate di `getStaticProps` buat content semi-dinamis.
- Hindari npm install paket berat kalau gak perlu (hemat bundle size).

🛣️ Routing Dinamis di Next.js

Kadang kita pengen bikin halaman yang isinya dinamis, contohnya `/blog/apa-itu-nextjs`. Nah, di Next.js lo tinggal pakai kurung kotak.

Pages Router

```
pages/
└── blog/
    └── [slug].js   # URL: /blog/apa-itu-nextjs
```

```
// pages/blog/[slug].js
export async function getStaticPaths() {
  const posts = await fetchPosts();
  const paths = posts.map(post => ({ params: { slug: post.slug } }));
  return { paths, fallback: true };
}

export async function getStaticProps({ params }) {
  const post = await fetchPost(params.slug);
  return { props: { post } };
}
```

🧰 Layouting Modular (App Router)

Kalau pakai App Router, lo bisa bikin layout per bagian. Gak perlu import layout di tiap halaman manual.

```
// app/layout.js
export default function RootLayout({ children }) {
  return (
    <html lang="id">
      <body>
        <Header />
        {children}
        <Footer />
```

```
      </body>
    </html>
  );
}
```

Mau layout yang beda untuk halaman admin? Bisa banget tinggal taruh di `app/admin/layout.js`. Clean dan scalable!

🔐 Middleware (Next.js 12+)

Middleware tuh kayak penjaga gerbang. Bisa lo pake buat:

- Redirect user kalau belum login
- Deteksi lokasi/region
- Blokir bot atau IP tertentu

Contoh:

```
// middleware.js
import { NextResponse } from 'next/server';

export function middleware(request) {
  const loggedIn = request.cookies.get('token');
  if (!loggedIn) {
    return NextResponse.redirect(new URL('/login', request.url));
  }
  return NextResponse.next();
}

export const config = {
  matcher: ['/dashboard/:path*'],
};
```

🔗 Next.js + Backend atau CMS

Pengen ambil data dari backend Laravel, Express, atau CMS kayak Strapi, Sanity, atau WordPress?

Gampang:

```
const res = await fetch('https://api.mycms.com/posts');
const data = await res.json();
```

Bisa di-fetch di `getStaticProps` atau langsung client-side (misalnya dengan SWR atau React Query).

🌙 Dark Mode & Theme Switcher

Dark mode gampang banget implementasinya pake Tailwind + Next.js.

```
// tailwind.config.js
module.exports = {
  darkMode: 'class',
  ...
}
```

Lalu tambahin toggle button:

```
<button onClick={() => document.documentElement.classList.toggle('dark')}>
  Toggle Theme
</button>
```

📦 Optimisasi Lanjutan

- **Bundle Analyzer** buat liat ukuran file:

```
npm install @next/bundle-analyzer
```

- **Code Splitting** otomatis, tapi bisa juga manual dengan dynamic import:

```
const KomponenBerat = dynamic(() => import('../components/KomponenBerat'), {
  loading: () => <Loading />,
});
```

- **Caching Headers** via Next.js config atau CDN (diatur di vercel.json atau middleware).

💬 Frequently Asked (FAQ)

“Gimana cara handle form di Next.js?”

Pake state biasa + `fetch` ke API route lo:

```
const handleSubmit = async () => {
  await fetch('/api/contact', {
    method: 'POST',
    body: JSON.stringify({ nama, email }),
  });
};
```

“Bisa Next.js tanpa typescript?”

BISA BANGET. Tapi kalau lo mau project yang skalabel, disaranin pake TypeScript.

“Gimana bedain SSR sama CSR (client-side rendering)?”

- **SSR**: data udah ada waktu page dimuat.
- **CSR**: data dimuat setelah komponen render (pakai `useEffect`, fetch client-side).

🔐 Integrasi Auth (Login, Logout, Protected Route)

Autentikasi di Next.js bisa macem-macem. Lo bisa:

- Pakai **NextAuth.js** buat social login (Google, GitHub, dsb)
- Atau bikin manual pakai JWT, cookies, dan API route

Contoh Manual (Simpel):

```
// pages/api/login.js
export default function handler(req, res) {
  const { email, password } = req.body;
  if (email === 'admin@tngdemy.id' && password === 'rahasia') {
    res.setHeader('Set-Cookie', 'token=123abc; Path=/');
    res.status(200).json({ message: 'Login berhasil!' });
  } else {
    res.status(401).json({ message: 'Gagal login' });
  }
}
```

Logout:

```
// pages/api/logout.js
export default function handler(req, res) {
  res.setHeader('Set-Cookie', 'token=; Max-Age=0; Path=/');
```

```
  res.status(200).json({ message: 'Logged out' });
}
```

🧪 Fitur Baru di Next.js 14+ yang Keren

Baru-baru ini, Next.js nambahin fitur makin dewa:

- **Turbopack**: pengganti Webpack, build & dev server super cepat
- **Partial Prerendering (PPR)**: render sebagian dulu, lanjut sisanya pas udah ada data
- **Server Actions (experimental)**: panggil fungsi server langsung dari komponen tanpa fetch manual

```
// app/page.js
'use server'

export async function simpanData(formData) {
  // langsung di-handle di server
}
```

Masih experimental, tapi bakal jadi masa depan Next.js.

📁 Struktur Folder yang Rapi ala Pro Developer

Supaya gak berantakan dan scalable, lo bisa pakai struktur kayak gini:

```
app/
├── layout.js
├── page.js
├── components/
│   └── Navbar.jsx
```

```
│   └── Footer.jsx
├── lib/
│   └── fetcher.js
├── styles/
│   └── globals.css
├── api/
│   └── auth/
│       └── login.js
├── dashboard/
│   └── layout.js
│   └── page.js
public/
├── images/
```

Kuncinya: **modular**, **terpisah antar tanggung jawab**, dan **mudah maintain**.

⚔️ SSG vs SSR vs ISR vs CSR

Metode	Kapan Render?	Contoh Cocok Buat
SSG	Saat build	Blog statis, halaman “Tentang”
SSR	Saat request masuk	Dashboard dinamis, data sering berubah
ISR	Saat build + regenerate	Artikel yang update tiap 5 menit
CSR	Setelah halaman dimuat	Komentar, chat, data realtime

Mau kombinasi? Bisa. Next.js ngasih fleksibilitas banget. Satu halaman bisa pakai SSR, satu lagi SSG. Bebas!

📤 Upload File di Next.js (Tanpa Library Ribet)

Next.js bisa handle upload image atau file lain langsung via API route.

```
// pages/api/upload.js
import formidable from 'formidable';
```

```
export const config = {
  api: {
    bodyParser: false,
  },
};

export default async function handler(req, res) {
  const form = new formidable.IncomingForm();
  form.uploadDir = './public/uploads';
  form.keepExtensions = true;
  form.parse(req, (err, fields, files) => {
    if (err) return res.status(500).json({ message: 'Gagal upload' });
    return res.status(200).json({ files });
  });
}
```

Di frontend, lo tinggal bikin form `<input type="file" />` dan kirim pakai `FormData`.

📝 Catatan Akhir buat yang Lagi Ngulik

- Gak usah nunggu jago buat mulai, mulai dulu sambil jalan.
- Next.js itu bener-bener "React yang di-upgrade", jadi makin banyak belajar React, makin gampang pahamin Next.js.
- Dokumentasi mereka itu emas. Kalo bingung, buka nextjs.org/docs.
- Kodingan error? Jangan panik. Console log aja, atau buka DevTools. Semua developer pernah ngalamin. Bahkan yang udah expert.