

Pengenalan Next.js

Tujuan Pembelajaran

1. Memahami konsep dasar Next.js dan fitur-fiturnya.
2. Mampu membuat aplikasi web dengan server-side rendering (SSR) dan static site generation (SSG).
3. Memahami penggunaan routing, API routes, dan optimasi gambar di Next.js.
4. Mampu membangun aplikasi web yang dinamis dan kompleks dengan Next.js.

Alat dan Bahan

1. Komputer dengan sistem operasi Windows, macOS, atau Linux.
2. Node.js dan npm (Node Package Manager) terinstal.
3. Text editor seperti Visual Studio Code.
4. Browser modern (Chrome, Firefox, Edge, dll.)

1. Apa itu Next.js?

Next.js adalah **framework open-source** yang dibangun di atas React. Next.js memungkinkan pengembang untuk membuat aplikasi web dengan **server-side rendering (SSR)** dan **static site generation (SSG)**. Framework ini dirancang untuk mengatasi beberapa keterbatasan React, seperti masalah kecepatan loading, SEO, dan keamanan.

Sejarah Next.js

Next.js pertama kali dirilis oleh **Vercel** (sebelumnya dikenal sebagai Zeit) pada tahun 2016. Framework ini dikembangkan untuk menyelesaikan masalah yang sering dihadapi oleh React, seperti:

- **Loading time yang lambat.**
- **Masalah SEO** karena React hanya melakukan rendering di sisi client.
- **Keamanan dan aksesibilitas.**

Pada tahun 2019, Google mulai berkontribusi pada proyek Next.js, dan sejak itu, banyak perusahaan besar seperti **Walmart**, **Apple**, **Netflix**, dan **Starbucks** menggunakan Next.js untuk membangun aplikasi web mereka.

2. Fitur Utama Next.js

Next.js menawarkan beberapa fitur utama yang membuatnya menjadi pilihan populer di kalangan pengembang:

1. Server-Side Rendering (SSR)

Next.js memungkinkan rendering komponen React di server sebelum mengirimkannya ke client. Ini memberikan beberapa keuntungan:

- **Loading time yang lebih cepat:** Pengguna melihat konten lebih cepat karena HTML sudah di-render di server.
- **SEO yang lebih baik:** Mesin pencari dapat mengindeks konten dengan lebih efektif karena HTML sudah dihasilkan di server.
- **Aksesibilitas:** Aplikasi lebih mudah diakses oleh pengguna dengan keterbatasan atau perangkat yang lambat.

```
1  // `pages` directory
2
3  export async function getServerSideProps() {
4    const res = await fetch('https://...')
5    const projects = await res.json()
6
7    return { props: { projects } }
8  }
9
10 export default function Dashboard({ projects }) {
11   return (
12     <ul>
13       {projects.map((project) => (
14         <li key={project.id}>{project.name}</li>
15       ))}
16     </ul>
17   )
18 }
```

Ketika pengguna mengunjungi halaman ini, Next.js akan merender komponen `HomePage` di server dan mengirimkan HTML yang sudah di-render ke client.

2. Static Site Generation (SSG)

Next.js memungkinkan pembuatan situs statis (static site) pada waktu build. Ini sangat berguna untuk situs dengan konten yang jarang berubah, seperti blog atau portofolio.

```

1  // `pages` directory
2
3  export async function getStaticProps() {
4    const res = await fetch(`https://...`)
5    const projects = await res.json()
6
7    return { props: { projects } }
8  }
9
10 export default function Index({ projects }) {
11   return projects.map((project) => <div>{project.name}</div>)
12 }

```

Fungsi `getStaticProps` akan dijalankan pada waktu build, dan data yang diambil akan digunakan untuk menghasilkan halaman statis.

3. Automatic Code Splitting

Next.js secara otomatis memisahkan kode JavaScript untuk setiap halaman. Ini berarti hanya kode yang diperlukan untuk halaman tertentu yang akan dimuat, sehingga meningkatkan kecepatan loading.

4. Image Optimization

Next.js menyediakan fitur optimasi gambar yang memungkinkan pengembang untuk:

- **Mengoptimalkan ukuran gambar** untuk berbagai perangkat.
- **Memilih format gambar terbaik** (seperti WebP) secara otomatis.
- **Lazy-loading**: Gambar hanya dimuat ketika diperlukan.

Contoh Penggunaan Image Component

```

1  import Image from 'next/image'
2
3  export default function Page() {
4    return (
5      <Image
6        src="https://s3.amazonaws.com/my-bucket/profile.png"
7        alt="Picture of the author"
8        width={500}
9        height={500}
10     />
11    )
12 }

```

5. Built-in Routing

Next.js menyediakan sistem routing berbasis file. Setiap file di dalam direktori pages akan otomatis menjadi route.

Contoh Routing di Next.js

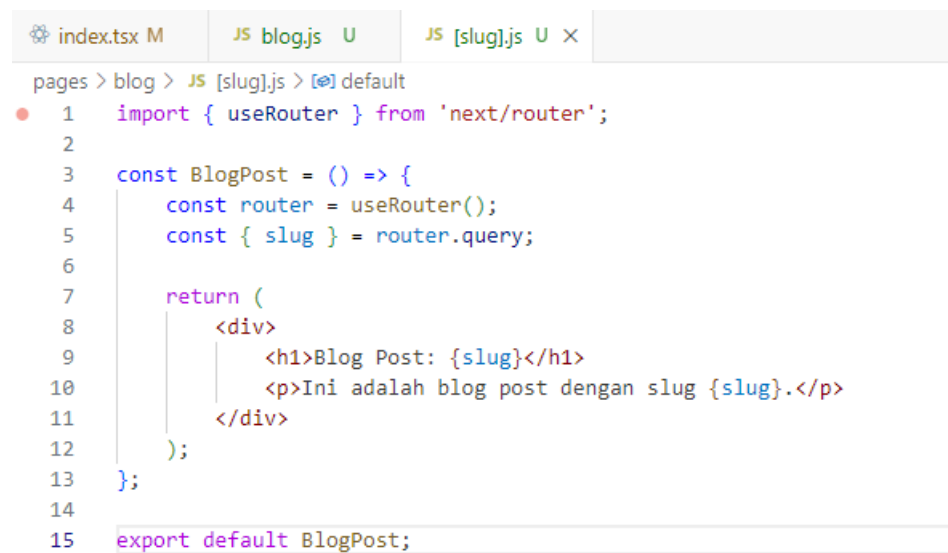
/pages

/index.js // Halaman Home

/about.js // Halaman About

/blog/[slug].js // Halaman Blog dengan Dynamic Route

Contoh Dynamic Route



```
index.tsx M JS blog.js U JS [slug].js U X
pages > blog > JS [slug].js > default
1 import { useRouter } from 'next/router';
2
3 const BlogPost = () => {
4   const router = useRouter();
5   const { slug } = router.query;
6
7   return (
8     <div>
9       <h1>Blog Post: {slug}</h1>
10      <p>Ini adalah blog post dengan slug {slug}</p>
11    </div>
12  );
13 };
14
15 export default BlogPost;
```

Jika pengguna mengunjungi /blog/my-first-post, Next.js akan menampilkan halaman dengan slug my-first-post.

6. API Routes

Next.js memungkinkan pembuatan API routes yang dapat digunakan untuk mengambil data dari backend atau API eksternal.

Contoh API Route

A screenshot of a code editor with multiple tabs at the top: 'index.tsx M', 'JS blog.js U', 'JS [slug].js U', 'JS products.js ...\api U X', and 'JS products.js'. The active tab is 'JS products.js'. The breadcrumb navigation shows 'pages > api > JS products.js > handler'. The code in the editor is as follows:

```
1 export default async function handler(req, res) {
2   const response = await fetch('https://fakestoreapi.com/products');
3   const products = await response.json();
4
5   res.status(200).json(products);
6 }
```

3. Kesimpulan

Next.js adalah framework yang powerful untuk membangun aplikasi web modern dengan fitur-fitur seperti **Server-Side Rendering (SSR)**, **Static Site Generation (SSG)**, **Automatic Code Splitting**, dan **Built-in Routing**. Dengan Next.js, pengembang dapat membuat aplikasi web yang cepat, SEO-friendly, dan mudah dikelola.

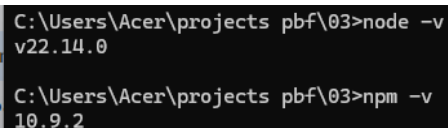
Kelebihan Next.js

- **Performansi tinggi:** Dengan SSR dan SSG, aplikasi menjadi lebih cepat.
- **SEO yang baik:** Konten di-render di server, sehingga mudah diindeks oleh mesin pencari.
- **Fleksibilitas:** Mendukung dynamic routes, API routes, dan optimasi gambar.

Langkah-langkah Praktikum

1. Persiapan Lingkungan

1. Pastikan Node.js dan npm sudah terinstal di komputer Anda. Anda dapat memeriksanya dengan menjalankan perintah berikut di terminal atau command prompt:

A screenshot of a terminal window with a black background and white text. It shows two commands and their outputs:

```
C:\Users\Acer\projects pbf\03>node -v
v22.14.0

C:\Users\Acer\projects pbf\03>npm -v
10.9.2
```

2. Buat direktori baru untuk proyek Next.js Anda
3. Inisialisasi proyek Next.js dengan menjalankan perintah berikut: Perhatikan bahwa **App Router** belum digunakan

```

C:\Users\Acer\projects pbf\0003>npx create-next-app@latest .
✓ Would you like to use TypeScript? ... No / Yes
✓ Would you like to use ESLint? ... No / Yes
✓ Would you like to use Tailwind CSS? ... No / Yes
✓ Would you like your code inside a `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to use Turbopack for `next dev`? ... No / Yes
✓ Would you like to customize the import alias (`@/*` by default)? ... No / Yes
Creating a new Next.js app in C:\Users\Acer\projects pbf\0003.

```

4. Jalankan aplikasi Next.js dengan perintah:

```

C:\Users\Acer\projects pbf\0003>npm run dev

> 0003@0.1.0 dev
> next dev

  ▲ Next.js 15.1.7
  - Local:      http://localhost:3000
  - Network:    http://192.168.74.55:3000

  ✓ Starting...
  ✓ Ready in 1980ms
  ○ Compiling / ...

```

Aplikasi akan terbuka di browser pada alamat <http://localhost:3000>.

2. Membuat Halaman dengan Server-Side Rendering (SSR)

1. Buka file pages/index.tsx di text editor Anda.
2. Ganti kode di dalamnya dengan kode berikut untuk membuat halaman sederhana:

```

pages > index.tsx > default
1 | import React from 'react';
2 |
3 | const HomePage = () => {
4 |   return (
5 |     <div>
6 |       <h1>Selamat Datang di Website Saya!</h1>
7 |       <p>Ini adalah halaman utama.</p>
8 |     </div>
9 |   );
10| };
11|
12| export default HomePage;

```

3. Simpan file dan lihat perubahan di browser. Anda akan melihat halaman utama dengan teks "Selamat Datang di Website Saya!".

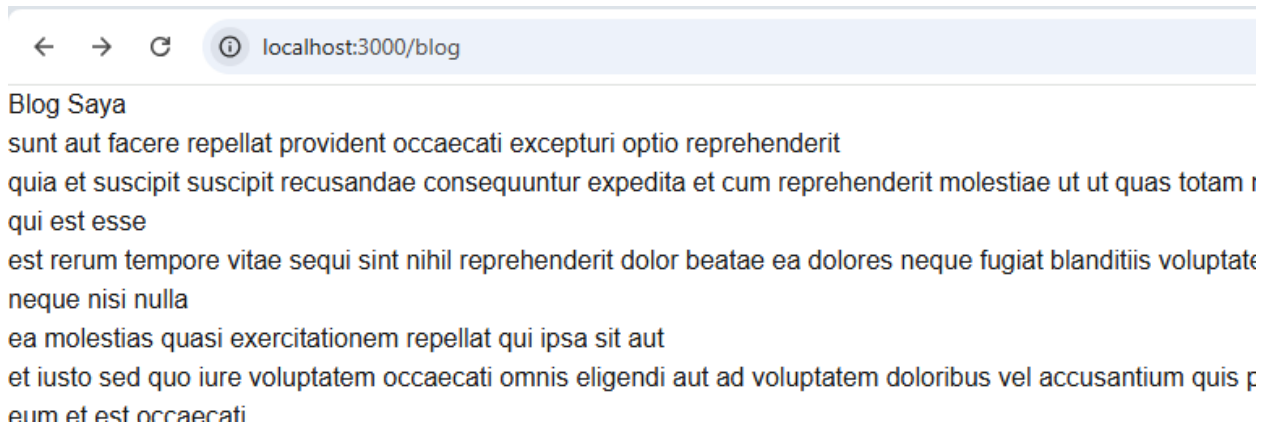
3. Menggunakan Static Site Generation (SSG)

1. Buat file baru di direktori **pages** dengan nama **blog.js**.

2. Tambahkan kode berikut untuk membuat halaman blog dengan SSG:

```
index.tsx M JS blog.js U X
pages > JS blog.js > ...
1  import React from 'react';
2
3  const Blog = ({ posts }) => {
4    return (
5      <div>
6        <h1>Blog Saya</h1>
7        {posts.map((post) => (
8          <div key={post.id}>
9            <h2>{post.title}</h2>
10           <p>{post.body}</p>
11         </div>
12       )]}
13     </div>
14   );
15 };
16
17 export async function getStaticProps() {
18   const res = await fetch('https://jsonplaceholder.typicode.com/posts');
19   const posts = await res.json();
20
21   return {
22     props: {
23       posts,
24     },
25   };
26 }
27
28 export default Blog;
```

3. Simpan file dan buka <http://localhost:3000/blog> di browser. Anda akan melihat daftar post yang diambil dari API eksternal.

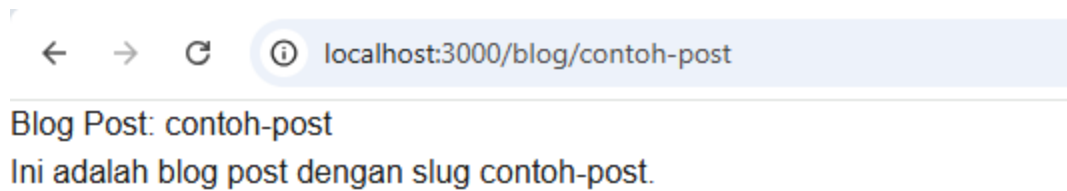


4. Menggunakan Dynamic Routes

1. Buat direktori baru di **pages** dengan nama **blog**.
2. Buat file di dalam direktori blog dengan nama **[slug].js**
3. Tambahkan kode berikut untuk membuat halaman dinamis berdasarkan slug:

```
index.tsx M JS blog.js U JS [slug].js U ×
pages > blog > JS [slug].js > default
1 import { useRouter } from 'next/router';
2
3 const BlogPost = () => {
4   const router = useRouter();
5   const { slug } = router.query;
6
7   return (
8     <div>
9       <h1>Blog Post: {slug}</h1>
10      <p>Ini adalah blog post dengan slug {slug}</p>
11    </div>
12  );
13 };
14
15 export default BlogPost;
```

4. Simpan file dan buka <http://localhost:3000/blog/contoh-post> di browser. Anda akan melihat halaman yang menampilkan slug dari URL.



5. Menggunakan API Routes

1. Pastikan terdapat direktori di pages dengan nama api.
2. Buat file di dalam direktori **api** dengan nama **products.js**.
3. Tambahkan kode berikut untuk membuat API route yang mengembalikan daftar produk:

```
index.tsx M JS blog.js U JS [slug].js U JS products.js ...api U X JS products.js
pages > api > JS products.js > handler
1 export default async function handler(req, res) {
2   const response = await fetch('https://fakestoreapi.com/products');
3   const products = await response.json();
4
5   res.status(200).json(products);
6 }
```

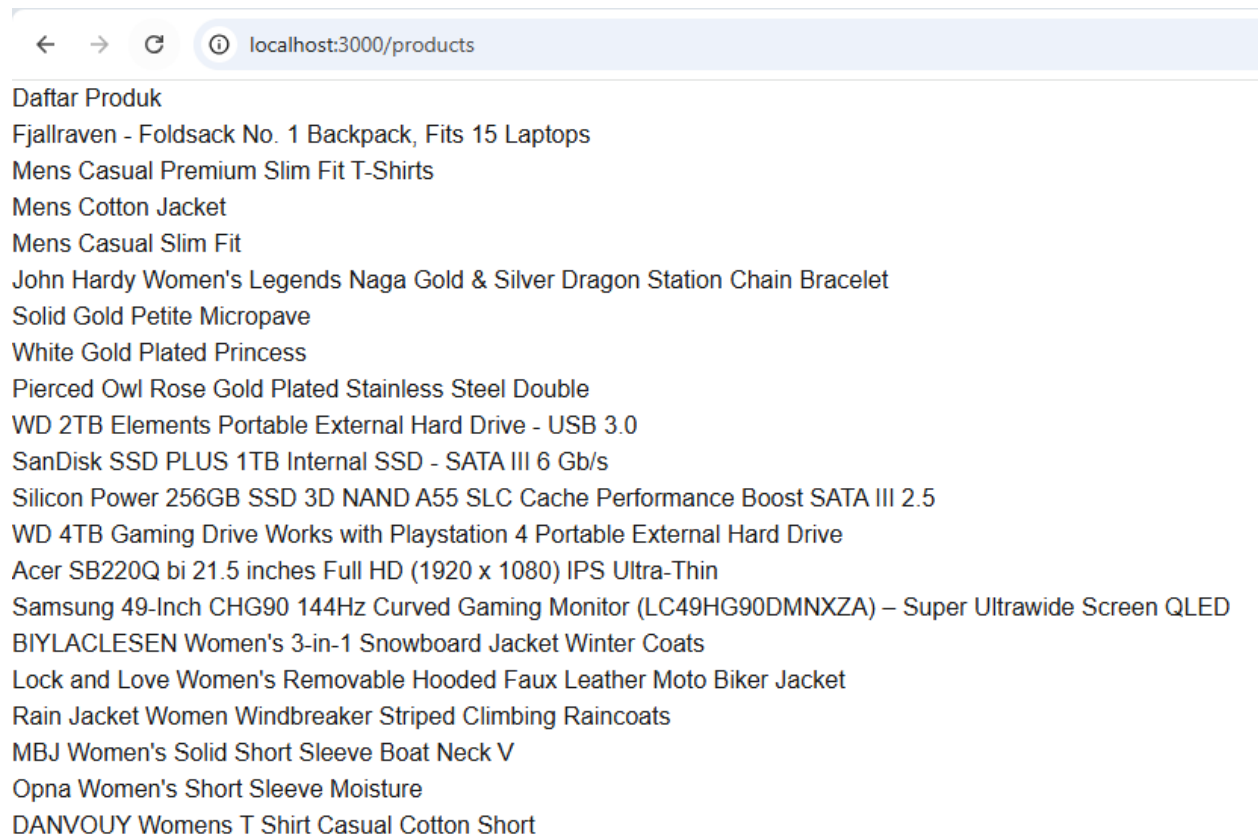
4. Buat file baru di **pages** dengan nama **products.js** untuk menampilkan daftar produk:

index.tsx M JS blog.js U JS [slug].js U JS products.js ...api U JS products.js pages U X

pages > JS products.js > [0] default

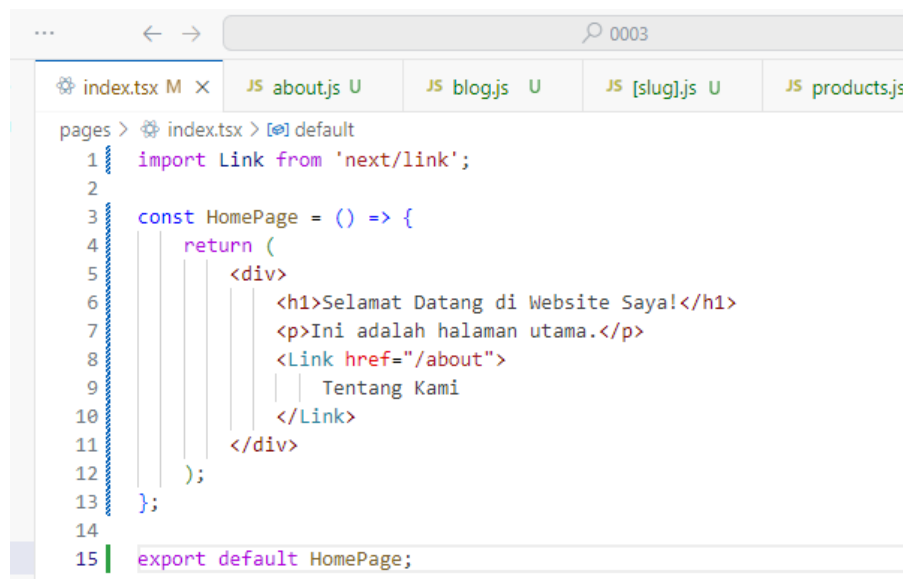
```
1 import { useState, useEffect } from 'react';
2
3 const ProductList = () => {
4   const [products, setProducts] = useState([]);
5
6   useEffect(() => {
7     const fetchProducts = async () => {
8       const response = await fetch('/api/products');
9       const products = await response.json();
10      setProducts(products);
11    };
12
13    fetchProducts();
14  }, []);
15
16  return (
17    <div>
18      <h1>Daftar Produk</h1>
19      <ul>
20        {products.map((product) => (
21          <li key={product.id}>{product.title}</li>
22        ))}
23      </ul>
24    </div>
25  );
26 };
27
28 export default ProductList;
```

5. Simpan file dan buka <http://localhost:3000/products> di browser. Anda akan melihat daftar produk yang diambil dari API route.



6. Menggunakan Link Component

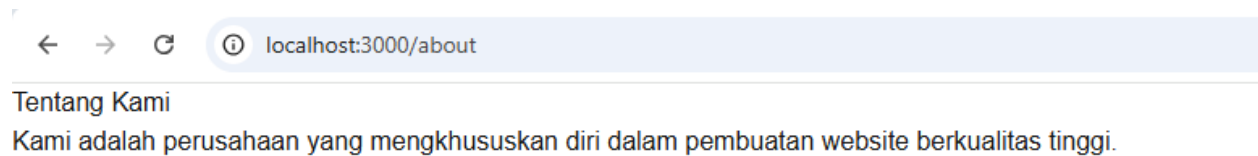
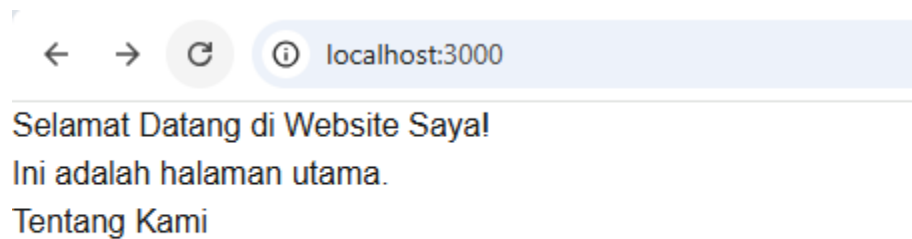
1. Buka file pages/index.tsx dan tambahkan modif dengan kode berikut untuk membuat link ke halaman lain:



2. Buat file baru di pages dengan nama about.js untuk halaman "Tentang Kami":

```
index.tsx M JS about.js U X JS blog.js U JS [slug].js U JS products.js ...api U JS products.js pages U
pages > JS about.js > default
1  const AboutPage = () => {
2      return (
3          <div>
4              <h1>Tentang Kami</h1>
5              <p>Kami adalah perusahaan yang mengkhususkan diri dalam pembuatan website berkualitas tinggi.</p>
6          </div>
7      );
8  };
9
10 export default AboutPage;
```

3. Simpan file dan buka <http://localhost:3000> di browser. Klik link "Tentang Kami" untuk navigasi ke halaman tentang.



Tugas

1. Buat halaman baru dengan menggunakan **Static Site Generation (SSG)** yang menampilkan daftar pengguna dari API <https://jsonplaceholder.typicode.com/users>.
2. Implementasikan **Dynamic Routes** untuk menampilkan detail pengguna berdasarkan ID.
3. Buat **API route** yang mengembalikan data cuaca dari API eksternal (misalnya, OpenWeatherMap) dan tampilkan data tersebut di halaman front-end.

Kesimpulan

Dalam praktikum ini, Anda telah mempelajari dasar-dasar Next.js, termasuk **Server-Side Rendering (SSR)**, **Static Site Generation (SSG)**, **Dynamic Routes**, **API Routes**, dan penggunaan **Link Component**. Next.js adalah framework yang powerful untuk membangun aplikasi web modern dengan performa tinggi dan pengalaman pengguna yang baik.