**Beginner: Getting Started with Next.js**

1. **Introduction to Next.js**
   * Learn the basics: Next.js is a React framework that enables server-side rendering and static site generation.
   * Next.js Documentation
2. **Setting up a Next.js App**
   * Install Next.js:

bash

Copy code

npx create-next-app@latest

* + Learn folder structure:
    - pages/: Each file represents a route.
    - public/: For static assets like images.
    - styles/: For CSS files.

1. **Routing in Next.js**
   * **File-based routing**: Pages are automatically served based on the pages/ directory.
   * Dynamic routes:
     + For pages like /post/[id], use a file like post/[id].js for dynamic routes.
   * Catch-all routes (...params): Handle multiple dynamic segments.
2. **Basic Page Rendering**
   * **Static Generation (SSG)**: Pages are generated at build time.

js

Copy code

export async function getStaticProps() {

return { props: { data } };

}

* + **Server-Side Rendering (SSR)**: Pages are generated on each request.

js

Copy code

export async function getServerSideProps() {

return { props: { data } };

}

1. **Styling**
   * Global CSS: Import CSS in \_app.js.
   * CSS Modules: Locally scoped CSS (e.g., component.module.css).
   * Styled JSX: Write scoped CSS inside components.

**Intermediate: Advanced Features and Optimization**

1. **API Routes**
   * Create API endpoints in the pages/api/ directory. Each file exports a handler function.

js

Copy code

export default function handler(req, res) {

res.status(200).json({ name: 'API Route' });

}

1. **Pre-rendering & Data Fetching**
   * **Incremental Static Regeneration (ISR)**: Regenerate static pages after certain intervals.

js

Copy code

export async function getStaticProps() {

return {

props: { data },

revalidate: 10, // Rebuild page every 10 seconds

};

}

1. **Image Optimization**
   * Use the next/image component for responsive image loading and optimization.

js

Copy code

import Image from 'next/image';

<Image src="/me.png" alt="Me" width={500} height={500} />

1. **Custom Document and App**
   * Customize \_document.js for server-side rendered HTML, e.g., for fonts.
   * Customize \_app.js for global settings (e.g., layout, styles).
2. **Middleware**
   * Middleware runs before the request is processed, allowing you to add logic like authentication.

js

Copy code

import { NextResponse } from 'next/server';

export function middleware(req) {

// Add custom logic before request is completed

return NextResponse.next();

}

1. **API with GraphQL**
   * Integrate GraphQL by setting up Apollo Client or similar for data fetching.

bash

Copy code

npm install @apollo/client graphql

1. **Custom 404 Page**
   * Create a custom 404 page by adding a 404.js file in the pages/ folder.

**Advanced: Full Optimization and Real-World Applications**

1. **Advanced Data Fetching Strategies**
   * Combine **SSG** with **ISR** for large-scale apps.
   * Use **react-query** for client-side data fetching with caching and revalidation.
2. **Authentication and Authorization**
   * Use **NextAuth.js** or implement JWT-based authentication.
   * Protect pages with middleware or getServerSideProps.
3. **Internationalization (i18n)**
   * Configure Next.js for multiple languages and locales.

js

Copy code

// next.config.js

module.exports = {

i18n: {

locales: ['en', 'fr', 'de'],

defaultLocale: 'en',

},

};

1. **Custom Server**
   * Create a custom server with Express or Fastify for advanced use cases.

js

Copy code

const express = require('express');

const next = require('next');

const app = next({ dev: true });

const handle = app.getRequestHandler();

app.prepare().then(() => {

const server = express();

server.get('\*', (req, res) => handle(req, res));

server.listen(3000);

});

1. **Performance Optimization**
   * **Code Splitting**: Automatically split code to reduce bundle size.
   * **Bundle Analysis**: Use the @next/bundle-analyzer to optimize performance.
   * **Pre-fetching**: Next.js prefetches pages in the background for faster load times.
   * **Static Assets**: Serve assets via a CDN by setting up the public directory efficiently.
2. **Serverless Deployment**
   * Deploy to platforms like Vercel, Netlify, or AWS Lambda with zero configuration for serverless functions.
3. **TypeScript**
   * Add TypeScript support by installing the necessary dependencies.

bash

Copy code

npm install --save-dev typescript @types/react

* + Create a tsconfig.json file, and Next.js will automatically detect TypeScript files.

1. **WebSockets for Real-Time Features**
   * Integrate WebSockets (e.g., Socket.IO) for real-time features like chat or notifications.
2. **Testing**
   * Unit testing with **Jest** and **React Testing Library**:

bash

Copy code

npm install jest @testing-library/react @testing-library/jest-dom

* + End-to-end testing with **Cypress** or **Playwright**.

1. **Edge Functions**

* Deploy and run functions closer to users at the edge for low-latency responses.

**Conclusion**

* **Practice with Real Projects**: Implementing advanced features like SSR, ISR, and API routes in real-world applications will push your understanding.
* **Stay Updated**: Next.js releases frequent updates. Follow their official channels and explore new features regularly.

Top of Form

Bottom of Form