Series 3: Advanced Routing & API Layer — a key area for full-stack and front-end interviews with Next.js.

🔷 Series 3: Advanced Routing & API Layer

> **Goal: Master dynamic routing, catch-all routes, API endpoints, and middleware — all frequently asked in interviews.**

🔥 Must-Know Topics (Common in Interviews)

- **Dynamic Routes (`[id].js`)**
- **Catch-All Routes (`[...slug].js`)**
- **API Routes (`pages/api/`)**
- **Middleware (auth, redirects)**
- **Custom Error Pages**
- **Redirects & Rewrites**
- **Protected Routes**

---

1. How do dynamic routes work in Next.js?

✅ **Interview Answer:**
Next.js allows dynamic segments in routes using brackets like `[id].js`. These files match dynamic URL values and can be used with `getStaticPaths` or `getServerSideProps`.

🧪 **Example:**

```
// pages/blog/[id].js
export async function getStaticProps({ params }) {
  const post = await fetchPost(params.id);
```

```
  return { props: { post } };
}
```

🧠 **Analogy:**
Dynamic routes are like templates for letters — just insert a name, and you're good to go.

---

2. What is a catch-all route and when do you use it?

✅ **Interview Answer:**
A catch-all route uses `[...slug].js` to match multiple segments (e.g., `/docs/a/b/c`). It's useful for building nested docs, breadcrumbs, or dynamic paths.

🧪 **Example:**

```
// pages/docs/[...slug].js
export default function DocsPage({ params }) {
  return <p>Showing path: {params.slug.join('/')}</p>;
}
```

🧠 **Analogy:**
Think of it as a vacuum — it catches anything passed down the route path.

---

3. What are API routes in Next.js and how do you create them?

✅ **Interview Answer:**
Next.js allows you to build backend logic directly inside `pages/api/`. Each file becomes an endpoint. These can handle authentication, DB operations, or proxying.

🧪 **Example:**

```
// pages/api/hello.js
export default function handler(req, res) {
  res.status(200).json({ message: 'Hello API' });
}
```

🧠 **Analogy:**
API routes are like mini Express endpoints — no need for a separate backend.

---

4. How do you handle custom error pages (404/500)?

✅ **Interview Answer:**
You can create `pages/404.js` and `pages/500.js` to show custom error messages for not found and server errors respectively.

🧪 **Example:**

```
// pages/404.js
export default function Custom404() {
  return <h1>Page Not Found</h1>;
}
```

🧠 **Analogy:**
Like putting a friendly sign when a store is closed or under maintenance.

---

5. What is Middleware in Next.js and what can it do?

✅ **Interview Answer:**
Middleware in Next.js runs before a request is completed. It can modify requests, redirect, or protect routes — all at the edge for speed.

🧪 **Example:**

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

export function middleware(req) {
  const isLoggedIn = req.cookies.get('auth');
  if (!isLoggedIn) {
    return NextResponse.redirect(new URL('/login', req.url));
  }
}
```

🧠 **Analogy:**
Middleware is like a security guard — checks if you can proceed or not.

---

6. What's the difference between redirect and rewrite in Next.js?

✅ **Interview Answer:**

- **Redirect: Changes the URL in the browser.**
- **Rewrite: Keeps the URL but fetches content from another path.**

🧪 **next.config.js Example:**

```
module.exports = {
  async redirects() {
    return [{ source: '/old', destination: '/new', permanent: true }];
  },
  async rewrites() {
    return [{ source: '/blog', destination: '/api/blog-handler' }];
  }
}
```

🧠 **Analogy:**
Redirect is like forwarding mail to a new address. Rewrite is like having your mail handled from the backend while still showing your old address.

---

7. How do you protect routes in Next.js?

✅ **Interview Answer:**
For client-side protection, check auth in a `useEffect`. For SSR, validate inside `getServerSideProps`. For edge-level auth, use middleware.

🧪 **Example (SSR auth):**

```
export async function getServerSideProps(context) {
  const { req } = context;
  const token = req.cookies.token;
  if (!token) {
    return { redirect: { destination: '/login', permanent: false } };
  }
  return { props: {} };
}
```

🧠 **Analogy:**
It's like checking for a ticket before letting someone enter a theater.

---

8. Can you use API Routes to handle form submissions or uploads?

✅ **Interview Answer:**
Yes. API routes can handle form POST requests or file uploads using libraries like `formidable` or `multer`.

🧪 **Example Snippet:**

```
// pages/api/contact.js
export default async function handler(req, res) {
  const { name, email } = req.body;
  // Save to DB
  res.status(200).json({ success: true });
}
```

🧠 **Analogy:**
Think of it like a form handler at the back office — gets your data, processes it, and responds.