Next.js 14 Basics - A Beginner's Guide

1. Installation

You can create a new Next.js project using:

```
npx create-next-app@latest my-next-app
```

Then start the development server:

```
cd my-next-app
npm run dev
```

Your app will be available at http://localhost:3000.

---

2. Project Structure

Your Next.js project will have the following structure:

```
my-next-app/
│── app/         # (New App Router) Pages and layouts
│── public/      # Static files
│── styles/      # CSS files
│── components/ # Reusable components
│── package.json
│── next.config.js
```

---

3. The App Router (`app/` Directory)

Next.js 14 uses **React Server Components** by default.

Example: Home Page

Create a homepage using:

```
// app/page.js
export default function Home() {
  return <h1>Welcome to Next.js 14!</h1>;
}
```

Routing

- `app/page.js` → `/`
- `app/about/page.js` → `/about`
- `app/blog/page.js` → `/blog`

4. Layouts

Use layouts to keep UI consistent across pages.

```
// app/layout.js
export default function RootLayout({ children }) {
  return (
    <html lang="en">
      <body>
        <nav>Navbar</nav>
        {children}
      </body>
    </html>
  );
}
```

5. Fetching Data

Fetching data on the server is built-in with `async/await`.

```
// app/posts/page.js
async function getPosts() {
  const res = await fetch("https://jsonplaceholder.typicode.com/posts");
  return res.json();
}

export default async function Posts() {
  const posts = await getPosts();
  return (
    <div>
      <h1>Blog Posts</h1>
      <ul>
        {posts.slice(0, 5).map((post) => (
          <li key={post.id}>{post.title}</li>
        ))}
      </ul>
    </div>
  );
}
```

6. API Routes

Create API endpoints inside `app/api/`.

```
// app/api/hello/route.js
export async function GET() {
  return Response.json({ message: "Hello, Next.js!" });
}
```

Now visit http://localhost:3000/api/hello to see the response.

7. Dynamic Routes

To create dynamic pages, use `[param]` in folder names.

```
// app/blog/[id]/page.js
export default function BlogPost({ params }) {
  return <h1>Post ID: {params.id}</h1>;
}
```

Visiting `/blog/123` will show `Post ID: 123`.

---

8. Styling in Next.js

Global CSS

Define global styles in `styles/globals.css`:

```
body {
  font-family: sans-serif;
}
```

CSS Modules (Scoped CSS)

```
/* components/Button.module.css */
.btn {
  background: blue;
  color: white;
  padding: 10px;
}
// components/Button.js
import styles from "./Button.module.css";

export default function Button() {
  return <button className={styles.btn}>Click me</button>;
}
```

---

9. Navigation (Client Side)

Use `next/link` for client-side navigation.

```
import Link from "next/link";

export default function Home() {
  return (
    <nav>
      <Link href="/">Home</Link>
      <Link href="/about">About</Link>
    </nav>
  );
}
```

---

10. Deploying Next.js

Deploy to Vercel (Easiest)

```
npm install -g vercel
vercel
```

Self-hosting

```
npm run build
npm start
```

🚀 Next Steps

- Learn about **Middleware** (for auth & redirects)
- Explore **Server Actions** (form submissions)
- Use **Edge Functions** for fast APIs
- Optimize performance with **ISR (Incremental Static Regeneration)**

This is a **quick crash course** on Next.js 14! 🎉 Let me know if you need more details. 😊