

Building Advanced React Server/Client Applications using Next.js

DevUp 2024

Contact Information:

Matt Henroid

Managing Principal Consultant – Daugherty Business Solutions

<https://www.linkedin.com/in/matthew-henroid/>

Thank you to our Sponsors!



Speaker Introduction

- Professional Developer, Architect and Consultant for 23 years
- 16 years at Daugherty Business Solutions
 - Managing Principal Consultant
 - Enterprise LOS Leader for Software Architecture & Engineering Practice
 - Support an organization of 600 engineers
 - Spend my free time building tools to make my job easier
- Contact me
 - <https://www.linkedin.com/in/matthew-henroid/>
- Code for presentation on Github
 - <https://github.com/mhenroid/devup2024-nextjs-demo>



Talk Overview

- Intro to Next.js
- Getting Started
- Understanding Project Structure
- Demo App intro
- Routing
- Rendering
- Data Fetching
- Optimizations
- Deploying
- MUI & Authentication



Introduction to Next.js

- Client / Server side framework for building full-stack web apps
- Build using React
- Store frontend and backend in a Node single project
- Abstracts configuration and tooling for bundling, compiling and more
- Supports CSR, SSR and SSG

What comes built-in?

- TypeScript integration
- ESLint support
- Fast refresh / hot-reloading
- File based routing / layouts
- Client + Server-Side Rendering
- Extended 'fetch' API (memoization, data caching, revalidation)
- Built-in CSS and JavaScript Bundling
- Support for CSS modules, Tailwind CSS and CSS-in-JS
- Image, font, script optimizations
- Static Site Generation
- Automatic Code Splitting
- API Proxying
- Internationalization support
- Way too many other things to mention on one page...

Getting Started

```
○ → devup2024-nextjs-demo git:(main) ✗ npx create-next-app@latest
✓ What is your project named? ... getting-started
✓ 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 to use `src/` directory? ... No / Yes
✓ Would you like to use App Router? (recommended) ... No / Yes
✓ Would you like to customize the default import alias (@/*)? ... No / Yes
Creating a new Next.js app in /Users/mrh0908/Code/DevUp2024/devup2024-nextjs-demo/getting-started.

Using npm.

Initializing project with template: app

Installing dependencies:
- react
- react-dom
- next

Installing devDependencies:
- typescript
- @types/node
- @types/react
- @types/react-dom
- eslint
- eslint-config-next

(#####) # idealTree:getting-started: timing idealTree:#root Completed in 5021ms
```

> cd <app-dir>

> npm run dev

Browse to <http://localhost:3000>

Get started by editing `src/app/page.tsx`

By  **Vercel**

NEXT.JS

Docs →

Find in-depth information
about Next.js features and API.

Templates →

Explore starter templates for Next.js.

Learn →

Learn about Next.js in an
interactive course with quizzes!

Deploy →

Instantly deploy your Next.js site
to a shareable URL with Vercel.



Next.js Project Structure

Project Structure

FOLDERS: DEVUP2024-NEXTJS-DEMO

- > demo-app
- ✓ getting-started
 - > .next
 - > node_modules
- ✓ public
 - 📁 next.svg
 - 📁 vercel.svg
- ✓ src/app
 - ★ favicon.ico
 - # globals.css
 - ⚙️ layout.tsx
 - # page.module.css
 - ⚙️ page.tsx
 - 🔗 .eslintrc.json
 - 🔗 .gitignore
 - TS next-env.d.ts
 - JS next.config.mjs
 - { } package-lock.json
 - { } package.json
 - 📄 README.md
 - tsconfig.json
 - 📄 README.md

File/Folder	
/public	Static assets
/src/app	App Router dynamic content
layout.tsx	Root layout (more to come)
page.tsx	Root page (more to come)
.eslintrc.json	Configuration file for ESLint
next-env.d.ts	TypeScript declaration file Next.js
next.config.mjs	Configuration file for Next.js
package.json	Project dependencies and scripts
tsconfig.json	Configuration file for TypeScript
/.next	Runtime storage for server, static assets and cached pages

Project Structure

FOLDERS: DEVUP2024-NEXTJS-DEMO

- > demo-app
- ✓ getting-started
 - > .next
 - > node_modules
- ✓ public
 - 📁 next.svg
 - 📁 vercel.svg
- ✓ src/app
 - ★ favicon.ico
 - # globals.css
 - ⚙️ layout.tsx
 - # page.module.css
 - ⚙️ page.tsx
 - 🔗 .eslintrc.json
 - 🔗 .gitignore
 - TS next-env.d.ts
 - JS next.config.mjs
 - { } package-lock.json
 - { } package.json
 - 📖 README.md
 - tsconfig.json
 - 📖 README.md

Other top-level files / folders

next.config.js	Configuration file for Next.js
package.json	Project dependencies and scripts
instrumentation.ts	OpenTelemetry and Instrumentation file
middleware.ts	Next.js request middleware
.env	Environment variables
.env.local	Local environment variables
.env.production	Production environment variables
.env.development	Development environment variables
.eslintrc.json	Configuration file for ESLint
.gitignore	Git files and folders to ignore
next-env.d.ts	TypeScript declaration file for Next.js
tsconfig.json	Configuration file for TypeScript
jsconfig.json	Configuration file for JavaScript

Project Structure

middleware.ts

FOLDERS: DEVUP2024-NEXTJS-DEMO

- > demo-app
- ✓ getting-started
 - > .next
 - > node_modules
- ✓ public
 - 📁 next.svg
 - 📁 vercel.svg
- ✓ src/app
 - ★ favicon.ico
 - # globals.css
 - ⚙️ layout.tsx
 - # page.module.css
 - ⚙️ page.tsx
- 🔗 .eslintrc.json
- 🔗 .gitignore
- TS next-env.d.ts
- JS next.config.mjs
- { } package-lock.json
- { } package.json
- 📖 README.md
- 📄 tsconfig.json
- 📖 README.md

```
1 // export { auth as middleware } from "@auth";
2 import { NextRequest, NextResponse } from "next/server";
3
4 export const middleware = async (request: NextRequest) => {
5   console.log("Logging from middleware");
6   return NextResponse.next();
7 };
8
9 export const config = {
10   // matcher: "",
11   // matcher: ["/test"],
12   matcher: [
13     /*
14      * Match all request paths except for the ones starting with:
15      * - api (API routes)
16      * - _next/static (static files)
17      * - _next/image (image optimization files)
18      * - favicon.ico (favicon file)
19      */
20     "/((?!api|_next/static|_next/image|favicon.ico).*)",
21   ],
22 };
```

Project Structure

FOLDERS: DEVUP2024-NEXTJS-DEMO

- > demo-app
- ✓ getting-started
 - > .next
 - > node_modules
- ✓ public
 - 📁 next.svg
 - 📁 vercel.svg
- ✓ src/app
 - ★ favicon.ico
 - # globals.css
 - 🌀 layout.tsx
 - # page.module.css
 - 🌀 page.tsx
- 🔗 .eslintrc.json
- 🔗 .gitignore
- TS next-env.d.ts
- JS next.config.mjs
- { } package-lock.json
- { } package.json
- 📖 README.md
- TS tsconfig.json
- 📖 README.md

Environment variables

📄 .env.local

```
1 DB_HOST=localhost
2 DB_USER=myuser
3 DB_PASS=mypassword
```

📄 .env

```
1 TWITTER_USER=nextjs
2 TWITTER_URL=https://twitter.com/$TWITTER_USER
```

JS pages/index.js

```
1 export async function getStaticProps() {
2   const db = await myDB.connect({
3     host: process.env.DB_HOST,
4     username: process.env.DB_USER,
5     password: process.env.DB_PASS,
6   })
7   // ...
8 }
```

Do not check your .env files into source control!

Set environment variables in OS for extra peace of mind

Project Structure

Environment variables

FOLDERS: DEVUP2024-NEXTJS-DEMO

- > demo-app
- ✓ getting-started
 - > .next
 - > node_modules
- ✓ public
 - 📁 next.svg
 - 📁 vercel.svg
- ✓ src/app
 - ★ favicon.ico
 - # globals.css
 - ⚙️ layout.tsx
 - # page.module.css
 - ⚙️ page.tsx
- 🔗 .eslintrc.json
- 🔗 .gitignore
- TS next-env.d.ts
- JS next.config.mjs
- { } package-lock.json
- { } package.json
- 📖 README.md
- tsconfig.json
- 📖 README.md

Environment variables only available in Server components

```
1 DB_HOST=localhost
2 DB_USER=myuser
3 DB_PASS=mypassword
```

Use *NEXT_PUBLIC_* to make available in Client components

```
NEXT_PUBLIC_ANALYTICS_ID=abcdefghijkl
```

```
1 import setupAnalyticsService from '../lib/my-analytics-service'
2
3 // 'NEXT_PUBLIC_ANALYTICS_ID' can be used here as it's prefixed by 'NEXT_PUBLIC_'
4 // It will be transformed at build time to `setupAnalyticsService('NEXT_PUBLIC_ANALYTICS_ID')`
5 setupAnalyticsService(process.env.NEXT_PUBLIC_ANALYTICS_ID)
6
7 function HomePage() {
8   return <h1>Hello World</h1>
9 }
10
11 export default HomePage
```

Project Structure

next.config.mjs

FOLDERS: DEVUP2024-NEXTJS-DEMO

- > demo-app
- ✓ getting-started
 - > .next
 - > node_modules
- ✓ public
 - 📁 next.svg
 - 📁 vercel.svg
- ✓ src/app
 - ★ favicon.ico
 - # globals.css
 - ⚙️ layout.tsx
 - # page.module.css
 - ⚙️ page.tsx
- 🔗 .eslintrc.json
- 🔗 .gitignore
- TS next-env.d.ts
- JS next.config.mjs
- { } package-lock.json
- { } package.json
- 📖 README.md
- 📄 tsconfig.json
- 📖 README.md

Numerous options to configure Next.js

<https://nextjs.org/docs/pages/api-reference/next-config-js>

Eslint, custom build dirs, headers, redirects, rewrites, source maps, compression, asset prefixes

We'll use later to configure MUI

```
1  /** @type {import('next').NextConfig} */
2  const nextConfig = {
3    modularizeImports: {
4      "@mui/icons-material": {
5        transform: "@mui/icons-material/{{member}}",
6      },
7    },
8  };
9
10 export default nextConfig;
```

Demo App

Next.js Demo

Sandbox

- [Server Side Error Test](#)
- [404 Test](#)
- [Get Time \(Server Component\)](#)
- [Get Time \(Client Component\)](#)
- [Get Time \(Client Component\) - with hydration error](#)

To Do App

- [To Do Client App](#)
- [To Do Composite App](#)

Next.js Demo

Todo Client App

Add a new todo

- item 1
- item 2
- item 3
- item 4

Next.js Demo

Todo Composite App

Add a new todo

- item 1
- item 2
- item 3
- item 4

Routing

Routing (App router)

- App Router (newer) vs. Page Router (older)

Routing Files

<code>layout</code>	Shared UI for a segment and its children
<code>page</code>	Unique UI of a route and make routes publicly accessible
<code>loading</code>	Loading UI for a segment and its children
<code>not-found</code>	Not found UI for a segment and its children
<code>error</code>	Error UI for a segment and its children
<code>global-error</code>	Global Error UI
<code>route</code>	Server-side API endpoint
<code>template</code>	Specialized re-rendered Layout UI
<code>default</code>	Fallback UI for Parallel Routes

Nested Routes

<code>folder</code>	Route segment
<code>folder/folder</code>	Nested route segment

Dynamic Routes

<code>[folder]</code>	Dynamic route segment
<code>[... folder]</code>	Catch-all route segment
<code>[[... folder]]</code>	Optional catch-all route segment

Route Groups

<code>(folder)</code>	Group routes without affecting routing
<code>_folder</code>	Opt folder and all child segments out of routing

Routing (App router)

```
demo-app-final
├── .next
├── node_modules
├── public
├── src
│   ├── app
│   │   ├── (dashboard)
│   │   │   ├── errortest
│   │   │   ├── gettimeclient
│   │   │   ├── gettimeclienterror
│   │   │   ├── gettimeserver
│   │   │   ├── todoclient
│   │   │   ├── page.tsx
│   │   │   ├── todocomposite
│   │   │   │   └── page.tsx
│   │   │   └── api/todos
│   │   │       ├── [id]
│   │   │       │   ├── route.ts
│   │   │       │   └── route.ts
│   │   │       ├── error.tsx
│   │   │       ├── layout.tsx
│   │   │       ├── page.tsx
│   │   │       └── serverActions.ts
│   │   ├── components
│   │   ├── lib
│   │   ├── styles
│   │   ├── theme
│   │   └── middleware.ts
```

Route group (ignored in URI)

Nested page

Dynamic API route /api/todos/1234

API route /api/todos handler

Error page

Root layout

Root page

Linking and Navigating

- Routing handled differently on server vs client
 - Server – uses code-splitting to split application into smaller bundles before sending to client
 - Client – Prefetches and caches route segments (browser doesn't reload the page)
- Methods
 - <Link> Component
 - useRouter hook (in client components)
 - redirect (in server components)
 - Native History API

Linking and Navigating

<Link>

```
1 import Link from 'next/link'
2
3 export default function Page() {
4   return <Link href="/dashboard">Dashboard</Link>
5 }
```

useRouter hook (client)

```
1 'use client'
2
3 import { useRouter } from 'next/navigation'
4
5 export default function Page() {
6   const router = useRouter()
7
8   return (
9     <button type="button" onClick={() => router.push('/dashboard')}>
10       Dashboard
11     </button>
12   )
13 }
```

redirect (server)

app/team/[id]/page.tsx

TypeScript



```
1 import { redirect } from 'next/navigation'
2
3 async function fetchTeam(id: string) {
4   const res = await fetch('https://...')
5   if (!res.ok) return undefined
6   return res.json()
7 }
8
9 export default async function Profile({ params }: { params: { id: string } }) {
10   const team = await fetchTeam(params.id)
11   if (!team) {
12     redirect('/login')
13   }
14
15   // ...
16 }
```

Error Handling

- 404
- Uncaught Exceptions (server)

error.tsx

app/dashboard/error.tsx

TypeScript



```
1  'use client' // Error boundaries must be Client Components
2
3  import { useEffect } from 'react'
4
5  export default function Error({
6    error,
7    reset,
8  }: {
9    error: Error & { digest?: string }
10   reset: () => void
11 }) {
12   useEffect(() => {
13     // Log the error to an error reporting service
14     console.error(error)
15   }, [error])
16
17   return (
18     <div>
19       <h2>Something went wrong!</h2>
20       <button
21         onClick={
22           // Attempt to recover by trying to re-render the segment
23           () => reset()
24         }
25       >
26         Try again
27       </button>
28     </div>
29   )
30 }
```

Route Handlers

- Use route.ts file
- Create custom request handlers (GET, POST, PUT, DELETE, etc.)
- NextRequest object passed to handler
 - An extension of JavaScript Request object

Available request handlers

```
1 export async function GET(request: Request) {}
2
3 export async function HEAD(request: Request) {}
4
5 export async function POST(request: Request) {}
6
7 export async function PUT(request: Request) {}
8
9 export async function DELETE(request: Request) {}
10
11 export async function PATCH(request: Request) {}
12
13 // If 'OPTIONS' is not defined, Next.js will automatically implement 'OPTIONS'
14 export async function OPTIONS(request: Request) {}
```

Using NextRequest and NextResponse

```
1 import { NextRequest, NextResponse } from "next/server";
2 import { createRepo } from "@lib/todo";
3
4 export async function GET(request: NextRequest) {
5   let todoRepository = createRepo();
6   let todos = await todoRepository.getTodos();
7
8   return NextResponse.json(todos);
9 }
10
11 export async function POST(request: NextRequest) {
12   let todoRepository = createRepo();
13   const data = await request.json();
14   await todoRepository.addTodo(data);
15   return NextResponse.json({ message: "created" }, { status: 201 });
16 }
17
```


Dynamic Routes

- When using dynamic routes, additional context data sent as second parameter

Example	URL	params
app/dashboard/[team]/route.js	/dashboard/1	{ team: '1' }
app/shop/[tag]/[item]/route.js	/shop/1/2	{ tag: '1', item: '2' }
app/blog/[...slug]/route.js	/blog/1/2	{ slug: ['1', '2'] }

```
1 import { NextRequest, NextResponse } from "next/server";
2 import { createRepo } from "@lib/todo";
3
4 export async function GET(
5   request: NextRequest,
6   context: { params: { id: string } }
7 ) {
8   let todoRepository = createRepo();
9   let todo = await todoRepository.getTodo(context.params.id);
10  if (todo) {--
11  }
12  return NextResponse.json({ message: "no such item" }, { status: 404 });
13 }
14
15 export async function DELETE(
16   request: NextRequest,
17   context: { params: { id: string } }
18 ) {
19   let todoRepository = createRepo();
20   await todoRepository.deleteTodo(context.params.id);
21   return NextResponse.json({ message: "deleted" }, { status: 200 });
22 }
23 }
```

Middleware

- Take control over requests before they are routed
- Use scenarios
 - Authentication and Authorization
 - Server-Side redirects
 - Path Rewriting
 - Logging and analytics
 - Feature flagging
- Use the matcher to customize routes that execute middleware

```
1  import { NextRequest, NextResponse } from "next/server";
2
3  export const middleware = async (request: NextRequest) => {
4    const { pathname, origin } = request.nextUrl;
5    console.log("Middleware: ", pathname);
6    if (pathname === "/redirect") {
7      return NextResponse.redirect(`${origin}/redirecttarget`);
8    }
9
10   return NextResponse.next();
11 };
12
13 export const config = {
14   // matcher: "",
15   // matcher: ["/test"],
16   matcher: [
17     /*
18      * Match all request paths except for the ones starting with:
19      * - api (API routes)
20      * - _next/static (static files)
21      * - _next/image (image optimization files)
22      * - favicon.ico (favicon file)
23      */
24     "/((?!api|_next/static|_next/image|favicon.ico).*)",
25   ],
26 };
27
```

Rendering

Server components (SSR)

Client components (CSR)

Composite components (hybrid)

Server Rendering / Components

- Server rendering is the default in Next.js
- Benefits
 - Data fetching on server
 - Security
 - Caching
 - Performance
 - Initial Page Load
 - Search Engine Optimization
 - Streaming
- Limitations
 - Only for non-interactive activities

```
export default async function Home() {  
  const date = new Date().toISOString();  
  return (  
    <>  
      Server Time: {date}  
    </>  
  );  
}
```

```
1  import { TodoItem, createRepo } from "@lib/todo";  
2  import TodoDelete from "../TodoDelete";  
3  
4  export default async function TodoList() {  
5    let todoRepository = createRepo();  
6    let todos = await todoRepository.getTodos();  
7  
8    return (  
9      <ul>  
10        {todos.map((todo: TodoItem, index: Number) => (  
11          <li key={index.toString()}>  
12            <div>  
13              <span>{todo.title} </span>  
14              <TodoDelete id={todo.id} />  
15            </div>  
16          </li>  
17        )]}  
18      </ul>  
19    );  
20  }  
21
```

Client Rendering / Components

- Add “use client” to top of file
 - Declares boundary between Server and Client Component modules
- Benefits
 - Supports interactivity
 - Access to Browser APIs

```
1  "use client";
2
3  import { useEffect, useState } from "react";
4
5  function getTime() {
6    return new Date().toISOString();
7  }
8
9  export default function Home() {
10    const [date, setDate] = useState<string>("default");
11
12    useEffect(() => {
13      setDate(getTime());
14
15      const interval = setInterval(() => {
16        setDate(getTime());
17      }, 1000);
18      return () => clearInterval(interval);
19    }, []);
20
21    return <>Client Time: {date}</>;
22  }
```

Composite Components

Server Component

```
1 import Navigation from "next/navigation";
2 import Autorefresh from "../autorefresh";
3
4 export default async function Home() {
5   const date = new Date().toJSON();
6   return (
7     <>
8       Server Time: {date}
9       <Autorefresh />
10    </>
11  );
12 }
```

Client Component

```
1 "use client";
2 import React, { useEffect, useState } from "react";
3 import { useRouter } from "next/navigation";
4
5 export default function TodoForm() {
6   const [isAutoRefresh, setIsAutoRefresh] = useState(true);
7   const router = useRouter();
8
9   let reload = () => {
10     if (isAutoRefresh) {
11       router.refresh();
12     }
13   };
14
15   useEffect(() => {
16     const interval = setInterval(() => {
17       reload();
18     }, 1000);
19     return () => clearInterval(interval);
20   }, [isAutoRefresh]);
21
22   return (
23     <>
24       <br />
25       <select
26         onChange={(e) => {
27           let shouldRefresh = e.target.value === "refresh";
28           setIsAutoRefresh(shouldRefresh);
29         }}
30       >
31         <option value="refresh">Auto refresh</option>
32         <option value="">No refresh</option>
33       </select>
34     </>
35   );
36 }
```

- Combine Server Component with child Client Components to support interactivity

Best Practices

- Start by building Server Components first
- Use Client Components only when you need interactivity or need to use client side APIs
- Build small Client Components and use within Server Components to add interactivity
- You can't use a Server Component within a Client Component

Hydration Issues

```
1  "use client";
2
3  import { useEffect, useState } from "react";
4
5  function getTime() {
6    return new Date().toJSON();
7  }
8
9  export default function Home() {
10    let date = getTime();
11
12    return (
13      <div>
14        <span>Client Time: {date}</span>
15        { /* <span suppressHydrationWarning>Client Time: {date}</span> */ }
16      </div>
17    );
18  }
19
```

Next.js Demo

Client Time: 2024-08-11T02:01:31.447Z

! 1 error ✕

1 of 1 error

Next.js (14.2.5) ✕

Unhandled Runtime Error

Error: Text content does not match server-rendered HTML.
See more info here: <https://nextjs.org/docs/messages/react-hydration-error>

Text content did not match. Server: "2024-08-11T02:01:30.648Z" Client: "2024-08-11T02:01:31.430Z"

Data Fetching

fetch API

Caching / Revalidation

Server Actions

Fetching on client or server?

- Methods to fetch data
 - 'fetch' API on server
 - ORMs or Database Clients on server
 - Route Handlers on server via client
 - Data fetching on client

Caching using 'fetch' on server

By default 'fetch' retrieves fresh data

```
1 export default async function Page() {
2   const data = await fetch('https://api.example.com/...').then((res) =>
3     res.json()
4   )
5
6   return '...'
7 }
```

On-demand revalidation by path

```
1 import { revalidatePath } from 'next/cache'
2
3 export async function createPost() {
4   // Mutate data
5   revalidatePath('/posts')
6 }
```

Cache individual requests with 'force-cache'

```
fetch('https://...', { cache: 'force-cache' })
```

Revalidate at timed intervals

```
fetch('https://...', { next: { revalidate: 3600 } })
```

On-demand revalidation by tag

```
1 export default async function Page() {
2   const res = await fetch('https://...', { next: { tags: ['collection'] } })
3   const data = await res.json()
4   // ...
5 }
```

```
3 import { revalidateTag } from 'next/cache'
4
5 export async function action() {
6   revalidateTag('collection')
7 }
```

Server Actions

- Asynchronous functions executed on the server
- Can be called in Server and Client components
- Client uses POST to execute server action

Next.js Demo

Todo Composite App

Add a new todo

- item 1
- item 2
- item 3
- item 4

```
1 "use client";
2 import React from "react";
3 import { deleteTodo } from "@app/serverActions";
4 import { useRouter } from "next/navigation";
5
6 export default function TodoForm({ id }: { id: string }) {
7   const router = useRouter();
8
9   const onClick = (id: string) => {
10     deleteTodo(id).then(() => {
11       router.refresh();
12     });
13   };
14
15   return (
16     <button
17       onClick={() => {
18         onClick(id);
19       }}
20     >
21     Delete
22     </button>
23   );
24 }
```

/src/appserverActions.ts

```
1 "use server";
2 import { createRepo, TodoItem } from "@lib/todo";
3
4 export async function addTodo(item: TodoItem) {
5   let todoRepository = createRepo();
6   await todoRepository.addTodo(item);
7 }
8
9 export async function deleteTodo(id: string) {
10   let todoRepository = createRepo();
11   await todoRepository.deleteTodo(id);
12 }
```

Optimization

Optimizations

- Images
 - Next.js extends `` element with features for automatic image optimization
 - Size optimization, prevent layout shift, faster page loads
- Metadata
- Package bundling
- Script Optimization
- Lazy loading
- Instrumentation
- Open Telemetry
- Static Assets

Deploying

Deploying

- Deployment options
 - Self-hosted on Node.js server
 - Docker image
 - Static HTML files
- Production Builds
 - Run 'npm run build'
 - Run 'npm run start' to run in prod mode

```
• → demo-app-final git:(dev) x npm run build
```

```
> my-app@0.1.0 build
> next build
```

```
▲ Next.js 14.2.5
- Environments: .env.local
```

```
Creating an optimized production build ...
```

```
✓ Compiled successfully
✓ Linting and checking validity of types
✓ Collecting page data
✓ Generating static pages (11/11)
✓ Collecting build traces
✓ Finalizing page optimization
```

Route (app)	Size	First Load JS
o /	6.94 kB	94 kB
o /_not-found	875 B	88 kB
f /api/todos	0 B	0 B
f /api/todos/[id]	0 B	0 B
o /gettimeclient	358 B	87.4 kB
o /gettimeclienterror	295 B	87.4 kB
o /gettimeserver	459 B	87.5 kB
o /redirecttarget	136 B	87.2 kB
o /todoclient	764 B	87.8 kB
o /todocomposite	776 B	87.9 kB
+ First Load JS shared by all	87.1 kB	
chunks/23-923f2b6a1a474833.js	31.5 kB	
chunks/fd9d1056-d81867e045c01ccb.js	53.7 kB	
other shared chunks (total)	1.88 kB	

```
f Middleware 26.8 kB
```

```
o (Static) prerendered as static content
f (Dynamic) server-rendered on demand
```


Dockerizing

- Use Docker example
<https://github.com/vercel/next.js/tree/canary/examples/with-docker>
- Add .dockerignore
- Add Dockerfile
- Add *output*: “*standalone*” to next.config.mjs or next.config.js
- `docker build -t devup2024/nextjs-demo-app .`
- `docker run -p 3000:3000 devup2024/nextjs-demo-app`

Integration with MUI

```
> npm install @mui/material @mui/icons-material @emotion/react @emotion/styled @fontsource/roboto
```

Create theme and ThemeProvider

Use theme provider in root layout.tsx

Update next.config.mjs to modularize icons

Wrapping Up

- Intro to Next.js
- Getting Started
- Understanding Project Structure
- Demo App intro
- Routing
- Rendering
- Data Fetching
- Optimizations
- Deploying
- MUI & Authentication



Q&A