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) x 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
```

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

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
- ① README.md
- stsconfig.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

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 ① 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
	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

```
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

 s tsconfig.json
README.md
```

middleware.ts

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

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 stsconfig.json

README.md

Environment variables

```
DB_HOST=localhost
DB_USER=myuser
DB_PASS=mypassword

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

```
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

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
- (i) README.md

Environment variables

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=abcdefghijk
```

```
import setupAnalyticsService from '../lib/my-analytics-service'

// 'NEXT_PUBLIC_ANALYTICS_ID' can be used here as it's prefixed by '
// It will be transformed at build time to `setupAnalyticsService('as setupAnalyticsService(process.env.NEXT_PUBLIC_ANALYTICS_ID)

function HomePage() {
   return <h1>Hello World</h1>
}

export default HomePage
```

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
- s tsconfig.json
- README.md

next.config.mjs

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

```
/** @type {import('next').NextConfig} */
const nextConfig = {
    modularizeImports: {
        "@mui/icons-material": {
        transform: "@mui/icons-material/{{member}}",
        },
    },
    };

export default nextConfig;
```

Demo App

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 Composite App



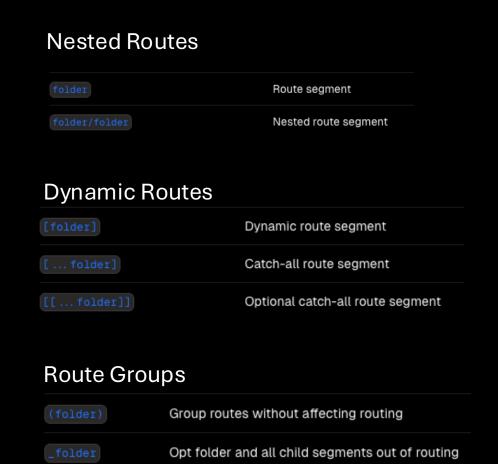
- item 1 Delete
- item 2 Delete
- item 3 Delete
- item 4 Delete

Routing

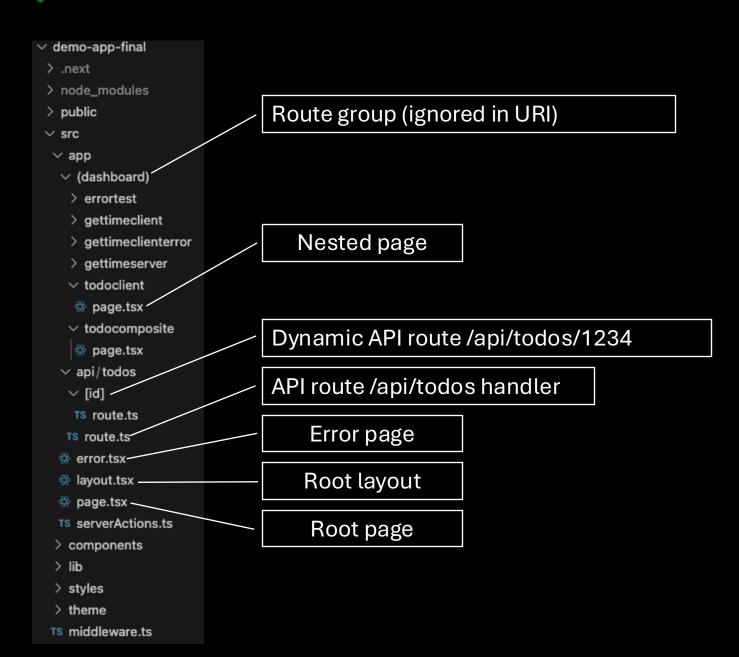
Routing (App router)

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

Routing Files Shared UI for a segment and its children Unique UI of a route and make routes publicly accessible Loading UI for a segment and its children Not found UI for a segment and its children not-found Error UI for a segment and its children Global Error UI Server-side API endpoint Specialized re-rendered Layout UI Fallback UI for Parallel Routes



Routing (App router)



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>

```
import Link from 'next/link'

export default function Page() {
   return <Link href="/dashboard">Dashboard</Link>
}
```

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 >
import { redirect } from 'next/navigation'

async function fetchTeam(id: string) {
 const res = await fetch('https://...')
 if (!res.ok) return undefined
 return res.json()

}

export default async function Profile({ params }: { params: { id: string oconst team = await fetchTeam(params.id)}

if (!team) {
 redirect('/login')

}

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ...

// ..

Error Handling

- 404
- Uncaught Exceptions (server)

error.tsx

TS app/dashboard/error.tsx



```
'use client' // Error boundaries must be Client Components
    import { useEffect } from 'react'
    export default function Error({
      error,
      reset,
      error: Error & { digest?: string }
      reset: () => void
11 }) {
      useEffect(() => {
        // Log the error to an error reporting service
        console.error(error)
      }, [error])
      return (
        <div>
          <h2>Something went wrong!</h2>
          <button
            onClick={
              // Attempt to recover by trying to re-render the segment
              () => reset()
            Try again
          </button>
        </div>
```

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

```
export async function GET(request: Request) {}

export async function HEAD(request: Request) {}

export async function POST(request: Request) {}

export async function PUT(request: Request) {}

export async function DELETE(request: Request) {}

export async function DELETE(request: Request) {}

// If `OPTIONS` is not defined, Next.js will automatically implement `OPTIONS' is not defined, Next.js will
```

Using NextRequest and NextResponse

```
import { NextRequest, NextResponse } from "next/server";
import { createRepo } from "@/lib/todo";

export async function GET(request: NextRequest) {
    tet todoRepository = createRepo();
    let todos = await todoRepository.getTodos();

    return NextResponse.json(todos);
}

export async function POST(request: NextRequest) {
    let todoRepository = createRepo();
    const data = await request.json();
    await todoRepository.addTodo(data);
    return NextResponse.json({ message: "created" }, { status: 201 });
}
```

Dynamic Routes

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



```
import { NextRequest, NextResponse } from "next/server";
     import { createRepo } from "@/lib/todo";
     export async function GET(
       request: NextRequest,
        context: { params: { id: string } }
        let todoRepository = createRepo();
        let todo = await todoRepository.getTodo(context.params.id);
10 >
       if (todo) {--
       return NextResponse.json({ message: "no such item" }, { status: 404 });
     export async function DELETE(
      request: NextRequest.
18
      context: { params: { id: string }
19
        let todoRepository = createRepo();
20
       await todoRepository.deleteTodo(context.params.id);
       return NextResponse.json({ message: "deleted" }, { status: 200 });
22
```

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

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

```
import { TodoItem, createRepo } from "@/lib/todo";
     import TodoDelete from "./TodoDelete";
     export default async function TodoList() {
       let todoRepository = createRepo();
       let todos = await todoRepository.getTodos();
       return (
         <l
           {todos.map((todo: TodoItem, index: Number) => (
10
             key={index.toString()}>
               <div>
                 <span>{todo.title} </span>
                 <TodoDelete id={todo.id} />
15
               </div>
16
             17
           ))}
18
         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

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

Composite Components

Server Component

```
import Navigation from "next/navigation";
import Autorefresh from "./autorefresh";

export default async function Home() {
    const date = new Date().toJSON();
    return (

    Server Time: {date}

    Autorefresh />
    );
}
```

 Combine Server Component with child Client Components to support interactivity

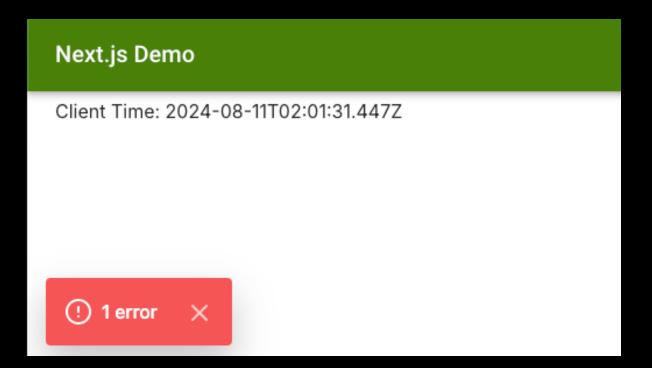
Client Component

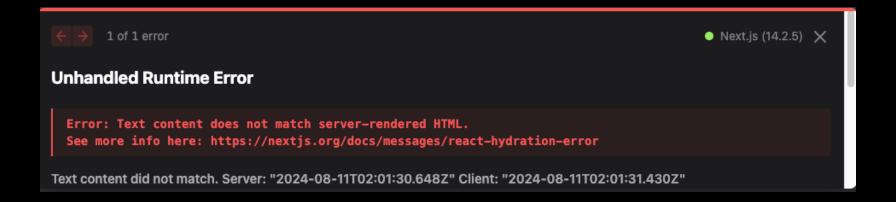
```
"use client";
import React, { useEffect, useState } from "react";
import { useRouter } from "next/navigation";
export default function TodoForm() {
 const [isAutoRefresh, setIsAutoRefresh] = useState(true);
 const router = useRouter();
  let reload = () => {
    if (isAutoRefresh) {
      router.refresh();
  };
 useEffect(() => {
    const interval = setInterval(() => {
      reload();
    }, 1000);
    return () => clearInterval(interval);
  }, [isAutoRefresh]);
  return (
      <br />
      <select
        onChange=\{(e) \Rightarrow \{
          let shouldRefresh = e.target.value === "refresh";
          setIsAutoRefresh(shouldRefresh);
        }}
        <option value="refresh">Auto refresh</option>
        <option value="">No refresh</option>
      </select>
```

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





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 }
```

Cache individual requests with 'force-cache'

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

Revalidate at timed intervals

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

On-demand revalidation by path

```
import { revalidatePath } from 'next/cache'

export async function createPost() {
    // Mutate data
    revalidatePath('/posts')
}
```

On-demand revalidation by tag

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

```
import { revalidateTag } from 'next/cache'

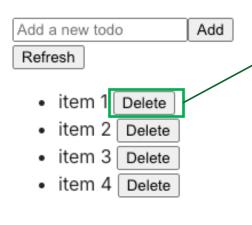
sexport async function action() {
   revalidateTag('collection')
}
```

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



```
"use client";
import React from "react";
import { deleteTodo } from "@/app/serverActions";
import { useRouter } from "next/navigation";
export default function TodoForm({ id }: { id: string }) {
  const router = useRouter();
 const onButtonClick = (id: string) => {
   deleteTodo(id).then(() => {
     router.refresh();
   });
  return (
    <but
     onClick={() => {
       onButtonClick(id);
     }}
     Delete
   </button>
```

/src/appserverActions.ts

```
"use server";
import { createRepo, TodoItem } from "@/lib/todo";

export async function addTodo(item: TodoItem) {
    let todoRepository = createRepo();
    await todoRepository.addTodo(item);
}

export async function deleteTodo(id: string) {
    let todoRepository = createRepo();
    await todoRepository.deleteTodo(id);
}
```

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
                                         Size
                                                  First Load JS
Route (app)
 0/
                                         6.94 kB
                                                          94 kB
                                         875 B
  o /_not-found
                                                          88 kB
  f /api/todos
                                         0 B
                                                            0 B
  f /api/todos/[id]
                                         0 B
                                                            0 B
                                         358 B
  o /gettimeclient
                                                        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
  (Static)
              prerendered as static content
   (Dynamic) server-rendered on demand
```

Dockerizing

- Use Docker example <u>https://github.com/vercel/next.js/tree/canary/examples/with-docker</u>
 - 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