



React with DOM

Darryl Ng



What is NextJS?

The React Framework for Production



React Framework for Production

React

- Library for building User Interfaces
- Need to make use of other libraries for feature rich application
 - Routing, Styling, Authentication, etc.

NextJS

- Package uses React for building user interface
- Built in features for full fledged production ready app
 - Routing, Styling, Authentication, Bundle Optimization, etc.
- No need to install additional package

Key features in NextJS?

Simplifies the process for building react application for production:

- File-based routing
- Pre-rendering
- API routes
- Support for CSS modules
- Authentication
- Dev & Prod build system

Pre-requisites

- HTML, CSS and JavaScript fundamentals
- ES6 + features
- React fundamentals
 - <https://legacy.reactjs.org/docs/getting-started.html>

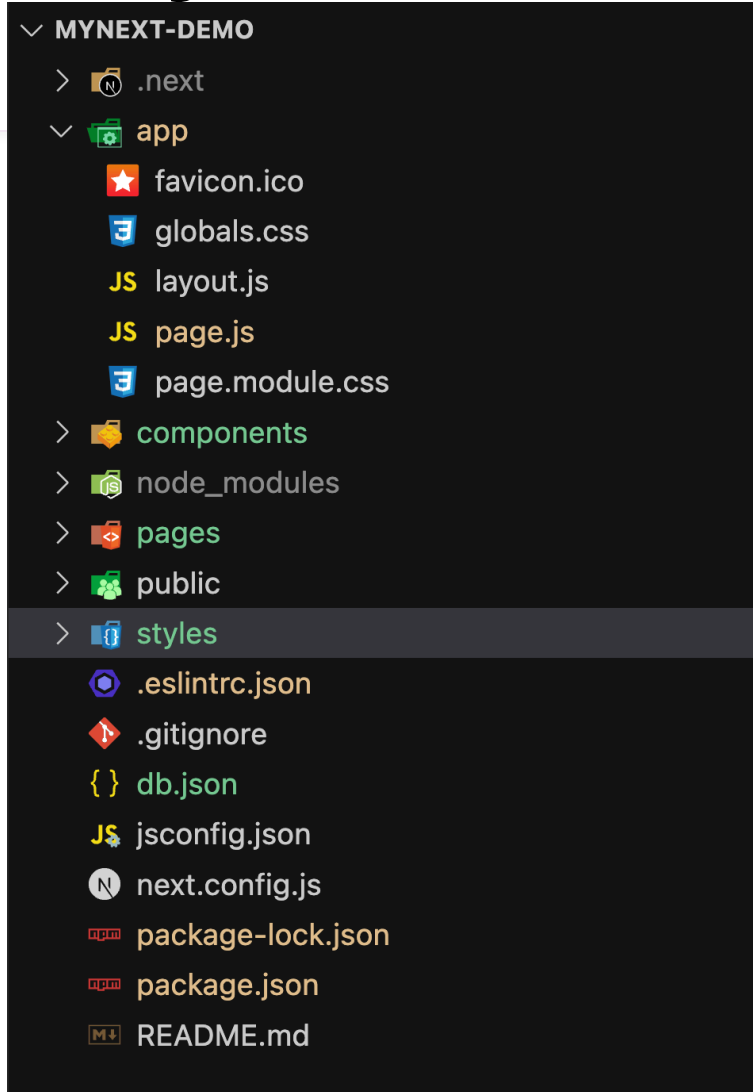
How To Create your First Basic NextJS app?

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

```
npm install next@latest react@latest react-dom@latest
```

```
npm run dev
```

Project Structure



<https://nextjs.org/docs/getting-started/project-structure>

Routing

React App

- Install 3rd party package
 - (react-router, react-router-dom)
- routes.js to configure routes
- Each route requires
 - Create a component file
 - Export the component
 - Import it in routes.js
 - Configure new routes with a path properly

```
import {Routes , Route } from "react-router-dom"
import Home from "../components/Home/Home"
import About from "../components/About/About"

function App(){
  return (
    <div className="App">
      <Routes>
        <Route path="/" component={<Home/> } />
        <Route path="/about" component={<About/> } />
      </Routes>
    </div>
  )
}

export default App
```


Routing in NextJS

- Route with Pages
- Nested routes
- Dynamic routes
- Catch-all routes
- Navigate from UI

Page Routing I

```
function FirstPage() {  
    return (<div>  
        <h1>First Page</h1>  
    </div>);  
}  
  
export default FirstPage;
```

Workshop - Page Routing

Estimate 5 minutes

Create a About page and Profile page with the similar code content as per FirstPage.js.

Page Routing II

- <http://localhost:3000/blog>
- <http://localhost:3000/blog/first>
- <http://localhost:3000/blog/second>

```
function Blog() {  
    return <h1>Blog Page</h1>;  
}
```

```
export default Blog;
```

```
function FirstBlog() {  
    return <h1>First blog page</h1>;  
}
```

```
export default FirstBlog;
```

```
function SecondBlog() {  
    return <h1>Second Blog Page</h1>;  
}
```

```
export default SecondBlog;
```

Page Routing III

- <http://localhost:3000/product>

- Product 1
- Product 2
- Product 3

- <http://localhost:3000/1>

- Product 1
Details

- <http://localhost:3000/2>

- Product 2
Details

Page Routing III (Code)

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

function ProductList( {productId = 100}) {
  return (
    <>
      <br />
      <Link href='/firstpage'>Home</Link>
      <br />
      <br />
      <Link href='/product/1'><h2>Product 1</h2></Link>
      <Link href='/product/2'><h2>Product 2</h2></Link>
      <Link href='/product/3' replace><h2>Product 3</h2></Link>
      <Link href={` /product/${productId}`}><h2>Product {productId}</h2></Link>
    </>
  )
}

export default ProductList;
```

Page Routing III (Code)

```
import { useRouter } from 'next/router';

function ProductDetail() {
  const router = useRouter();
  const productId = router.query.productId;

  return (<>
    <h1>Details about Product {productId}</h1>
    <br />
    <br />
  </>)
}

export default ProductDetail;
```

Page Routing IV (Catch All)

- <http://localhost:3000/doc/feature1/concept1>

```
import { useRouter } from "next/router";

function Doc() {
  const router = useRouter();
  const { params = [] } = router.query;
  console.log(params);

  if (params.length === 2) {
    return <h1>Viewing docs for feature {params[0]} and concept {params[1]}</h1>
  } else if (params.length === 1) {
    return <h1>Viewing docs for feature {params[0]}</h1>
  }

  return <h1>Docs Home Page</h1>;
}

export default Doc;
```


Bootstrap integration with NextJS

- Install bootstrap module
 - **npm install bootstrap**
- Import bootstrap css in **_app.js**
 - `import 'bootstrap/dist/css/bootstrap.min.css';`

Using and calling external resources

- Download db.json from
 - <https://github.com/darryl1975/nextjs-resources>
- Put db.json to project root folder
- npm install json-server
- npm run serve-json

Data Fetching - `getStaticProps`

- Data required for the page is available at build time during rendering
- Efficient/Performance due pre-rendered (for SEO)

getStaticProps

1. Function runs only on server side (not and never on client side).
2. Code written won't be included in JS bundle sent to browser.
3. Server side code can be written directly in **getStaticProps**.
4. Accessing file system (fs module) or querying database can be done inside the function.
5. Use of API keys to call API in getStaticProps won't be exposed to the browser.

getStaticProps (cont.)

6. Allowed only in pages, will not run in component file.
7. Used for pre-rendering and not client-side data fetching.
8. Returns an object. (Object contains a **props** key that contains an object.)
9. Runs at build time.
10. In development mode, getStaticProps runs on every request.

Data Fetching - `getStaticPaths`

- Data comes from a database/filesystem
- Efficient/Performance due pre-rendered (for SEO)

Fetching data from jsonPlaceholder

```
function PostList() {  
  return (<>  
    <div class="container">  
      <h1>List of Posts</h1>  
      <br />  
      <br />  
    </>)  
}  
  
export default PostList;
```

Data Fetching - getStaticProps

```
export async function getStaticProps() {  
  const response = await fetch('https://jsonplaceholder.typicode.com/posts');  
  
  const data = await response.json();  
  
  return {  
    props: {  
      // posts: data.slice(0, 3)  
      posts: data  
    }  
  }  
}
```


Completing the code in PostList() ...

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

function PostList({ posts }) {
  return (<>
    <div class="container">
      <h1>List of Posts</h1>
      <br />
      <br />

      {
        posts.map(post => {
          return (
            <div key={post.id}>
              <Link href={`posts/${post.id}`} passHref>
                <h2>{post.id} {post.title}</h2>
              </Link>
              <hr />
            </div>
          )
        })
      }
    </>)
}
```

Introducing nextUI (Replacing the code with nextUI)

npm install @nextui-org/react

```
import { Card, Row, Text } from '@nextui-org/react';

{
  posts.map(post => {
    return (
      <Card class="card text-bg-light mb-3">
        <Card.Body class="card-body">
          <Row key={post.id} justify="center" align="center">
            <Link href={`posts/${post.id}`} passHref>
              <Text h2 size={15} color="black">{post.id} {post.title}</Text>
            </Link>
          </Row>
        </Card.Body>
      </Card>
    )
  })
}
```

<https://nextui.org>

Post Details I

```
import { useRouter } from "next/router";

function Post({ post }) {
  const router = useRouter();

  if (router.isFallback) {
    return <h1>Loading...</h1>
  }

  return (<
    <h2> {post.id} {post.title}</h2>
    <p>{post.body}</p>
  </>);
}

export default Post;
```

Data Fetching - getStaticPaths

```
export async function getStaticPaths() {
  const response = await fetch('https://jsonplaceholder.typicode.com/posts');
  const data = await response.json();

  const paths = data.map(post => {
    return {
      params: {
        postId: `${post.id}`
      }
    }
  })

  return {
    paths,
    fallback: true
  }
}
```

Data Fetching - getStaticProps

```
export async function getStaticProps(context) {  
  const { params } = context;  
  const response = await fetch(`https://jsonplaceholder.typicode.com/posts/  
${params.postId}`);  
  
  const data = await response.json();  
  
  console.log(`Generating page for /posts/${params.postId}`);  
  
  return {  
    props: {  
      post: data  
    }  
  }  
}
```

Workshop – Page Routing IV

Estimate 15 minutes

- Retrieve list of products data from db.json and display on
 - <http://localhost:3000/products>
- Click on the product link and navigate to
 - <http://localhost:3000/products/{productid}>

Custom 404 Error Handling

- Name the page **404.js**
- Put inside the **pages** folder

```
function PageNotFound() {  
  return (<h1>404 Page with all the custom styling necessary</h1>);  
}  
  
export default PageNotFound
```

Event Handling

- Declare an event handler

```
const handleClick = () => {  
  console.log("Viewing products main page...");  
  router.push("/product");  
}
```

- Bind the event handler to the **onClick** event

Revisiting First Page I

```
import Link from "next/link";
import { useRouter } from "next/router";

function FirstPage() {
  const router = useRouter();

  const handleClick = () => {
    console.log("Viewing products main page...");
    router.push("/product");
  }

  ...
}

export default FirstPage;
```

Completing First Page

```
function FirstPage() {  
  ...  
  
  return (<div>  
    <h1>First Page</h1>  
    <br />  
    <Link href="/blog">Blog</Link>  
    <br />  
    <Link href="/product">Products</Link>  
    <br/>  
    <Link href="/posts">Posts</Link>  
    <br/>  
    <br/>  
    <button className="btn btn-primary" onClick={handleClick}>  
      Product List  
    </button>  
  </div>);  
}
```

Routing Summary

- Page based routing mechanism
 - Pages are associated with a route based on their file name
- Nested routes
 - Nested folder structure
 - Files automatically routed in the same way based on URL
- Dynamic routes
 - Created by creating square brackets to a page name

Routing Summary

- Catch all routes
 - Add three dots inside square brackets
 - Helpful when you want different URLs for same page layout
 - Working with pages where some route parameters are optional
- Link component to navigate on click of an element
- useRouter hook
 - router.push method to navigate programmatically
- Create custom 404 page

Building a User Listing Page

```
function UserList() {  
  return (<  
    <h1>User Listing</h1>  
    <br />  
  </>);  
}
```

```
export default UserList;
```

Retrieving data for User Listing Page

```
export async function getStaticProps() {  
  const response = await fetch('https://jsonplaceholder.typicode.com/users');  
  const data = await response.json();  
  
  console.log(data);  
  
  return {  
    props: {  
      users: data  
    }  
  };  
}
```

Displaying data on User Listing Page

```
function UserList({ users }) {  
  return (<>  
    <h1>User Listing</h1>  
    <br />  
    {  
      users.map(user => {  
        return (<div key={user.id}>  
          <p>{user.name}</p>  
          <p>{user.email}</p>  
          <br />  
        </div>)  
      })  
    }  
  </>);  
}
```

Reusable Components (for User Listing)

- Create a **components** folder
- Create the user.js code file with the following code

```
function User ({ user }) {  
  return (<  
    <p>{user.name}</p>  
    <p>{user.email}</p>  
  </>)  
}  
  
export default User;
```

<https://nextjs.org/learn/foundations/from-javascript-to-react/building-ui-with-components>

Using the Component in User Listing

```
import User from "../components/user";

function UserList({ users }) {
  return (<>
    <h1>User Listing</h1>
    <br />
    {
      users.map(user => {
        return (<div key={user.id}>
          <User user={user} />
          <br />
        </div>)
      })
    }
  </>);
}
```

Pre-rendering & Data Fetching Intro

Types of pre-rendering

1. Static generation✓
 1. With or without data
 2. Dynamic parameters when fetching data
2. Server-side rendering
3. Client-side data fetching
4. Combining pre-rendering with client-side data fetching

Static generation

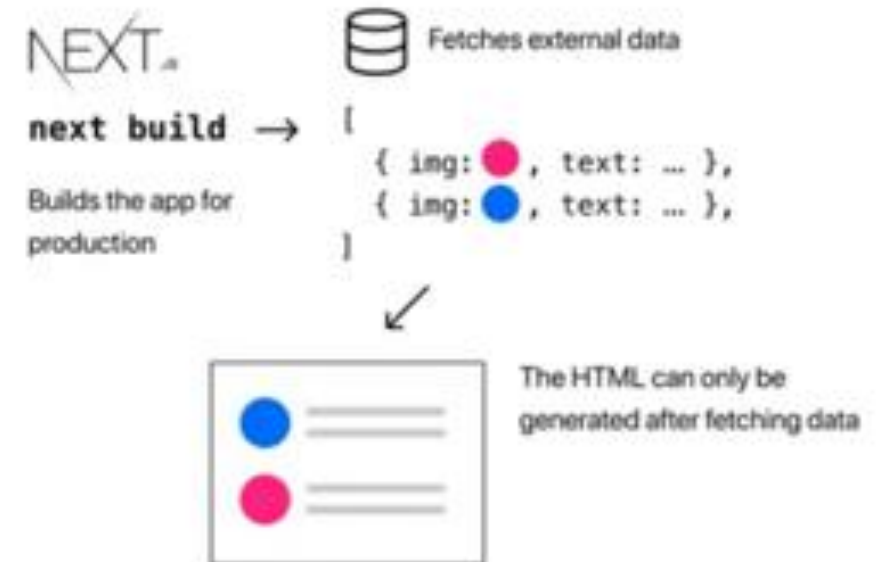
Static Generation without Data

For pages that can be generated without fetching external data at build time.



Static Generation with Data

For pages that can only be generated after fetching external data at build time.



Static generation (cont.)

- (Default) Method of pre-rendering HTML pages at build time.
- HTML with all data that makes up the content of the page are generated in advance when you build your application
- Pages built once, cached by CDN and served to client almost instantly
- E.g. blogs, products, documentation and marketing pages

Why Pre-rendering?

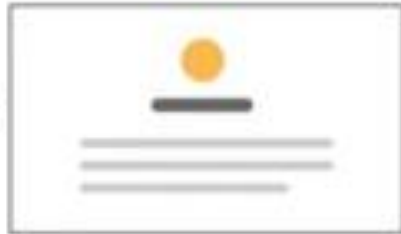
No Pre-rendering (Plain React.js app)

Initial Load:
App is not rendered



JS loads
→

Hydration: React components are initialized and App becomes interactive



Pre-rendering (Using Next.js)

Initial Load:
Pre-rendered HTML is displayed



JS loads
→

Hydration: React components are initialized and App becomes interactive



If your app has interactive components like `<Link />`, they'll be active after JS loads

Why Pre-rendering? (Cont.)

1. Pre-rendering improves performance
 - In React app, JavaScript has to be executed before rendering.
 - Fetch data using an external API, then render UI. (Wait time for user)
 - With pre-rendered page, HTML already generated and loads faster.
2. Pre-rendering helps with SEO
 - With React app, search engine hits the page and only sees `<div>` tag with id equals root.
 - Search engine hits pre-rendered page, and pages are indexed




Styling Intro


Styling is essential to building any web application

- Global styles
- Component styles
- SASS or SCSS
- CSS-in-JS solution

Global Styling

styles >  global.css > ...

```
98  */
99
100  a {
101    color: inherit;
102    text-decoration: none;
103  }
104
105  @media (prefers-color-scheme: dark) {
106    html {
107      color-scheme: dark;
108    }
109  }
110
111  h2 {
112    color:  orange;
113  }
```

pages >  _app.js > ...

```
1  import '../styles/globals.css'
2  import 'bootstrap/dist/css/bootstrap.min.css'
3
4  function MyApp({ Component, pageProps }) {
5    return <Component {...pageProps} />
6  }
7
8  export default MyApp
9  |
```


Workshop - Component Level Styling

Estimate 5 minutes

- Download the css resources from
 - <https://github.com/darryl1975/nextjs-resources>
- Copy the css resources to the project **styles** folder
- Import the css module in **about.js**

```
import styles from '../styles/About.module.css'
```

- Style your HTML code

```
function About() {  
  return (<>  
    <br />  
    <h1 className={styles.highlight}>About</h1>  
    <br />  
    </>);  
}
```

Styling Summary

- Global - Need to import CSS file within pages/_app.js
- Component - Supports CSS modules using [name].module.css naming convention
- SASS - install the sass package
- CSS-in-JS - inline styling and using styled components

Other NextJS features and support

App Layout

Head Component

Image Component

Absolute imports and module paths

Static HTML export

TypeScript support

Preview

Next Config File

Redirects

Environment variables

Authentication

Summary - Other NextJS features and support

- App layout in `_app.js` file
- Head component helps to dynamically manage document's head section
- Image component optimization
- Configure absolute imports and configure path aliases with `jsconfig.json`
- `next export` command exports app into static HTML
- Setup support for TypeScript
- Preview mode feature – helpful when working with CMS
- Next configuration file and configuring redirects

Fixing up the pieces..



Service layer - API call

```
import axios from 'axios';

const url = "http://127.0.0.1:4000/user";

export const getAllUsers = async (id) => {
  id = id || '';
  return await axios.get(`${url}/${id}`);
}

export const addUser = async (user) => {
  return await axios.post(url, user);
}

export const editUser = async (id, user) => {
  return await axios.put(`${url}/${id}`, user);
}

export const deleteUser = async (id) => {
  return await axios.delete(`${url}/${id}`);
}
```

Creating User Listing page...

```
import React, { useState } from 'react';
import { Table, Button } from '@nextui-org/react';
import { deleteUser } from '../../service/api';
import Link from 'next/link';
```

```
function AllUsers({ users }) {

  const [user, setUser] = useState(users);

  const deleteData = async (id) => {
    await deleteUser(id);
    window.location.reload(false);
  }

  return (
    <>
      ...
    </>
  )
}

export default AllUsers;
```

Using getStaticProps to retrieve users

```
export async function getStaticProps() {  
  const response = await fetch('http://localhost:4000/user');  
  
  const data = await response.json();  
  
  return {  
    props: {  
      users: data  
    },  
    revalidate: 30  
  }  
}
```



```
<div className='container'>
  <Link href="/user/AddUser" className="btn btn-success">New User</Link>
  <br />
  <Table className="table table-striped table-responsive">
    <Table.Header>
      <Table.Column>ID</Table.Column>
      <Table.Column>Name</Table.Column>
      <Table.Column>UserName</Table.Column>
      <Table.Column>Email</Table.Column>
      <Table.Column>Phone</Table.Column>
      <Table.Column></Table.Column>
    </Table.Header>
    <Table.Body items={user} >
      {user.map((item) => (
        <Table.Row key={item.id}>
          <Table.Cell className="overflow-auto">
            {item.id}
          </Table.Cell>
          <Table.Cell className="overflow-auto">
            {item.name}
          </Table.Cell>
          <Table.Cell className="overflow-auto">
            {item.username}
          </Table.Cell>
          <Table.Cell className="overflow-auto">
            {item.email}
          </Table.Cell>
          <Table.Cell className="overflow-auto">
            {item.phone}
          </Table.Cell>
          <Table.Cell className="overflow-auto">
            <Button className="btn btn-danger" onClick={() => deleteData(item.id)}>Delete</Button>
          </Table.Cell>
        </Table.Row>
      ))}
    </Table.Body>
  </Table>
</div>
```

References and Kickstart tutorials

- <https://github.com/suraj25809/React-CRUD-using-Json-Server>
- <https://efficientuser.com/2023/01/16/creating-a-todo-application-using-next-js/>

References and Kickstart tutorials

- **Understanding Server Components in React 18 and Next.js 13**

- <https://adhithiravi.medium.com/what-are-server-components-and-client-components-in-react-18-and-next-js-13-6f869c0c66b0>

References and Kickstart tutorials

- <https://www.youtube.com/watch?v=PcHY2Py6AQY>
- **Creating a Tabs component with Next.js**
 - <https://medium.com/backticks-tildes/creating-a-tabs-component-with-next-js-9c88bdc0e29e>
- **Next.js Unleashed: A Beginner's Guide to Start Developing Web Applications with Next.js**
 - <https://blog.devgenius.io/next-js-unleashed-a-beginners-guide-to-start-developing-web-applications-with-next-js-40ea4b230c0>

References and Kickstart tutorials

- **Mastering NextJS Routing: A Comprehensive Guide to Building Dynamic Web Applications with MUI Components**
 - <https://blog.devgenius.io/mastering-next-js-4cfff5a89882>
 - <https://blog.devgenius.io/mastering-next-js-7a31525d543f>
 - <https://blog.devgenius.io/mastering-next-js-mui-box-component-497d76a01528>



Thank you