Nextjs:

What is Next.js?

The React Framework for production.

**React:**

* Not quite possible to build a full feature rich application ready to be developed for production React is a library for building user interfaces
* React is a library for building user interfaces
* You have to make decisions on other features of the app like routing, styling, authentication etc.

**Next.js**

* A package that uses React for building user interfaces
* Loaded with a lot more features that enable you to build full fledged production ready applications. Features exactly like routing, styling, authentication, bundle optimization etc.
* There’s no need to install additional packages. Next.js provides everything for you.
* Opinions and conventions need to be followed to implement the above said features

Why learn Next.js?

Next.js simplifies the process of building a react application for production

1. File based routing
2. Pre-rendering
3. API routes
4. Support for CSS modules
5. Authentication
6. Development & production build system

**CSR vs SSR vs RSC:**

**Client-side Rendering (CSR):** Client-side scripting simply means running scripts, such as JavaScript, on the client device, usually within a browser. All kinds of scripts can run on the client side if they are written in JavaScript, because JavaScript is universally supported.

**SSR (Server-Side Rendering)**: Performs server-side rendering only during the initial page load. This means that other than the initial components, all other components are rendered in the browser when SSR is used. So, the browser has to download the JavaScript for the whole app just like in a Single-Page Application (SPA)

**RSC (React Server Components)**: Only some parts of the page are rendered on the server. The rest is rendered in the browser/client as needed. Additionally, every time you visit the page, it's going to render on the server.

There are three main ways that you can render in a Next. js application: pre-render with **server-side rendering (SSR)**, pre-render with **static site generation (SSG**), or updating/creating content at runtime with **incremental static regeneration (ISR**).

<https://dev.to/teyim/a-deep-dive-into-csr-ssr-ssg-and-isr-3513>

Pre-requisites

HTML, CSS and JavaScript fundamentals

ES6 + features

React Fundamentals

**Installation:**

[**https://nextjs.org/docs/getting-started/installation**](https://nextjs.org/docs/getting-started/installation)

Automatic Installation

We recommend creating a new Next.js app using create-next-app, which sets up everything automatically for you. To create a project, run:

Terminal>>

npx create-next-app@latest

npx create-next-app blog

npx create-next-app@13.1.6

yarn create next-app

yarn add [package-name@1.2.3](mailto:package-name@1.2.3)

<https://stackoverflow.com/questions/59996216/how-do-i-install-the-latest-version-of-next-js-with-yarn>

Details: <https://www.youtube.com/watch?v=y0ecd_bGKb4&ab_channel=CodeStepByStep>

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

**Run the application:** npm run dev or yarn run dev

**Need to know the folder file structure for starting**:

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

**How to Set port in next.js / default port 3000**

"scripts": {

"dev": "next dev -p 8080", // for dev

"start": "next start -p 8080" // for prod

},

With yarn, pnpm you can easily pass any arguments:  
yarn dev -p 8080 or yarn dev --port=8080

With npm using -- to pass arguments:  
npm run dev -- -p 8080

<https://stackoverflow.com/questions/60147499/how-to-set-port-in-next-js>

**Simple React Snippets**v1.2.8

Burke Holland

Vs code extension installed ,

ffc - Function Component

sfc - Stateless Function Component (Arrow function)

**Routing in a Next.js app:**

* Route with pages
* Nested routes
* Dynamic routes
* Catch-all routes
* Navigate from the UI
* Programmatically navigate between pages

**Pre-rendering & Data fetching Intro:**

What and why of pre-rendering

Types of pre-rendering

1. Static Generation

-- Without data

-- With data

-- Incremental Static Generation

-- Dynamic parameters when fetching data

2. Server-side Rendering

- Data fetching

\* Client-side data fetching

\* combining pre-rendering with client-side data fetching

Pre-rendering

React vs Next JS

By default, Next JS pre-renders every page in the application

What does pre-render mean?

Next JS generates HTML for each page in advance instead of having it all done by client-side JavaScript

Pre-render just means render in advance of sending to the browser.

Pre-rendering is done by default in a Next JS app

Why pre-render?

1. Pre-rendering improves performance.

\* In a react app, you need to wait for the JavaScript to be executed

\* Perhaps fetch data from an external API & then render the UI

\* There is a wait time for the user

\* With a pre-rendered page, the HTML is already generated and loads faster

2. Pre-rendering helps with SEO

\* If you are building a blog or an e-commerce site, SEO is a concern

\* With a React app, if the search engine hits your page, it only sees a div tag with id equal to root

\* If search engine hits a pre-rendered page though, all the content is present in the source code which will help index that page

\* If SEO is of concern for your app, pre-rendering is what you want

Pre-rendering Summery

Pre-rendering refers to the process of generating HTML with the necessary data for a page in your application.

Pre-rendering can result in better performance SEO.

Pre-rendering in Next JS

Next JS supports two forms of pre-rendering

* Static Generation. Static Site Generation (SSG)
* Server-side Rendering

**Static Generation:** A method of pre-rendering where the HTML pages are generated at build time

The HTML with all the data that makes up the content of the web page are generated in advance when you build your application

Recommended method to pre-render pages whenever possible

Page can be build once, cached by a CDN and served to the client almost instantly

Ex: Blog pages, e-commerce product pages, documentation and marketing pages

**Static Generation: -- How?**

Next JS, by default will pre-render every page in our app

The HTML for every page will automatically be statically generated when we build our application

Production Server:- An optimized build is created once and you deploy that build. You don’t make code changes on the go once it is deployed

Development Server:- We should be able to make changes in our code and we want that code to immediately reflect in the browser

For production builds, a page will be pre-rendered once when we run build commend

In development mode, the page is pre-rendered for every request you make

Does Next.js 13 use getStaticProps?

\*\* The new data fetching in Next. js 13 is built on top of the fetch() Web API and makes use of async / await in Server Components. Now, instead of using getServerSideProps() and getStaticProps() , all fetched data is static by default, meaning it's rendered at build time.

// JSONPlaceholder // Free fake API for testing and prototyping.

<https://jsonplaceholder.typicode.com/>

<> </>   /\*This is called fragment\*/

<https://nextjs.org/docs/app/building-your-application/routing/error-handling>

SSG - Static Site Generation (SSG)

SSR - server-side renders

ISR – incremental site regeneration

**Fetch Vs. Axios In Next.js For HTTP Requests: Which Is Better?**

**Install Tailwind CSS with Next.js**

<https://tailwindcss.com/docs/guides/nextjs>

**@tailwindcss/typography**

Beautiful typographic defaults for HTML you don't control.

The official Tailwind CSS Typography plugin provides a set of prose classes you can use to add beautiful typographic defaults to any vanilla HTML you don’t control, like HTML rendered from Markdown, or pulled from a CMS.

<https://tailwindcss.com/docs/typography-plugin#basic-usage>

Install the plugin from npm:

npm install -D @tailwindcss/typography

Font awesome icons in nextjs

First install the package:

npm i react-icons

Markdown/MDX with Next.js

If you're creating an information-dense website like documentation or a blog, you're probably considering using Markdown. Most developers are familiar with Markdown from GitHub and other online communities.

<https://nextjs.org/blog/markdown>

// Use gray-matter to parse the post metadata section

Install grey-matter:

npm i gray-matter

**The most popular component librarycomponent library for Tailwind CSS**

<https://daisyui.com/> > see components > install

//daisyUI adds component class names to Tailwind CSS so you can make beautiful websites faster than ever.

// admin panel tailwind free

<https://tailadmin.com/download>

<https://medevel.com/11-nextjs-dashboard/>

<https://github.com/kitloong/nextjs-dashboard>

**Tailwind CSS**

A utility-first CSS framework for rapidly building custom user interfaces.

**DaisyUI**

Free UI components plugin for Tailwind CSS

React Icons:

Include popular icons in your React projects easily with react-icons, which utilizes ES6 imports that allows you to include only the icons that your project is using.

<https://react-icons.github.io/react-icons/>

install: > npm i react-icons

**JSON Server:**

Get a full fake REST API with **zero coding** in **less than 30 seconds** (seriously)

<https://www.npmjs.com/package/json-server>

npm i json-server

Start JSON Server

json-server --watch db.json

Dummy api Live:

<https://dummy.restapiexample.com/>

**type aliases vs interfaces:**

// One major difference between type aliases vs interfaces are that interfaces are open and type aliases are closed. This means you can extend an interface by declaring it a second time. // In the other case a type cannot be changed outside of its declaration.

Types are better for working with functions, complex types, etc. **Interfaces work better with objects and method objects**.

**react-slick**

<https://www.npmjs.com/package/react-slick>

npm install react-slick --save

then install the type

npm i --save-dev @types/react-slick

**Also install slick-carousel for css and font**

npm install slick-carousel

// Import css files

import "slick-carousel/slick/slick.css";

import "slick-carousel/slick/slick-theme.css";

JavaScript Array map()

<https://www.w3schools.com/jsref/jsref_map.asp>

<https://www.geeksforgeeks.org/typescript-array-map-method/>

// language is TypeScript

    // Driver code

    var arr = [2, 5, 6, 3, 8, 9];

    // use of map() method

    var newArr = arr.map(function(val, index){

      // printing element

      console.log("key : ",index, "value : ",val\*val);

    })

**rc-table**

<https://www.npmjs.com/package/rc-table>

**rc-pagination**

<https://codepen.io/piyushpd139/pen/vYpZxxy>

<https://www.npmjs.com/package/rc-pagination>

**react-data-table-component**

<https://www.npmjs.com/package/react-data-table-component>

<https://www.youtube.com/watch?v=3oHUtG0cjfY&ab_channel=CodeWithYousaf>

<https://www.youtube.com/watch?v=QwQAQat_uT8&ab_channel=CodeWithYousaf>

**react-paginate // implemented**

<https://www.npmjs.com/package/react-paginate>

**date-fns**

date-fns gave us the power to work directly with date objects, without worrying about conversion or mutations. It's a real game changer for dates.

<https://www.npmjs.com/package/date-fns>

npm install date-fns --save

Formik:

Formik is a small library that helps you with the 3 most annoying parts:

1. Getting values in and out of form state
2. Validation and error messages
3. Handling form submission

npm install formik –save

<https://formik.org/docs/tutorial>

# React Reveal

[**React Reveal**](https://www.react-reveal.com/) is an animation framework for React. It's MIT licensed, has a tiny footprint and written specifically for React in ES6. It can be used to create various cool reveal on scroll animations in your application.

npm install react-reveal –save

<https://www.npmjs.com/package/react-reveal?activeTab=readme>

Fixing Error: [**Typescript react - Could not find a declaration file for module**](https://stackoverflow.com/questions/41462729/typescript-react-could-not-find-a-declaration-file-for-module-react-material)

Make sure to stop your react local server and start it again after doing the following:

1- Create .d.ts file manually, you just need to do the following:

2 - enter src folder

3 - create global.d.ts file

4 - declare modules in it like:

declare module 'module-name';

declare module 'react-reveal/Fade'

declare module 'react-reveal/Zoom'

<https://stackoverflow.com/questions/41462729/typescript-react-could-not-find-a-declaration-file-for-module-react-material>

**react-tabs  (Not checked)**

An accessible and easy tab component for ReactJS.

[**https://reactcommunity.org/react-tabs/**](https://reactcommunity.org/react-tabs/)

<https://www.npmjs.com/package/react-tabs>

<https://ej2.syncfusion.com/react/documentation/tab/how-to/tab-selection>

// For the gym website

npm i framer-motion react-countup react-scroll @types/react-scroll react-icons react-responsive swiper

**Swiper - is** the free and most modern mobile touch slider

<https://www.npmjs.com/package/swiper> <https://swiperjs.com/get-started>

<https://swiperjs.com/demos#pagination>

**React SwiperJs autoplay :**

<https://stackoverflow.com/questions/63052586/react-swiperjs-autoplay-not-making-the-swiper-to-auto-swipe>

for swiper version 8.3

imports should look like this

import { Autoplay, Pagination } from 'swiper';

import { Swiper, SwiperSlide } from 'swiper/react';

import "swiper/css";

import "swiper/css/pagination";

import "swiper/css/autoplay";

and your swiper react component, mine wasnt working on autoplay={true} so I added autoplay={{delay: 2000} anyways below is my whole swiper it will help you

<Swiper

modules={[Autoplay, Pagination]}

pagination={{clickable: true}}

slidesPerView={1}

autoplay={{

delay: 2000,

pauseOnMouseEnter: true,

disableOnInteraction: false

}}

loop

className='swiper-container'

>

**How to hide scrollbar on your element in TailwindCSS**

Source:

<https://dev.to/derick1530/how-to-create-scrollable-element-in-tailwind-without-a-scrollbar-4mbd>

Go to your global.css file styles/global.css and past this code:

//global index.css

@tailwind base;

@tailwind components;

@tailwind utilities;

// add the code bellow

@layer utilities {

/\* Hide scrollbar for Chrome, Safari and Opera \*/

.no-scrollbar::-webkit-scrollbar {

display: none;

}

/\* Hide scrollbar for IE, Edge and Firefox \*/

.no-scrollbar {

-ms-overflow-style: none; /\* IE and Edge \*/

scrollbar-width: none; /\* Firefox \*/

}

}

At this point we added ::-webkit-scrollbar to target the scrollbar style in Chrome,Safari, Edge and Opera.

no-scrollbar is the class that we are going to use for hidding the scrollbar.

Step II

Now use no-scrollbar to hide the scrollbar in your elements. Like this:

<div className="w-full h-42 overflow-y-scroll no-scrollbar">...</div>

**Aceternity UI**

<https://ui.aceternity.com/components/parallax-scroll>

Install dependencies

npm i framer-motion clsx tailwind-merge

<https://ui.aceternity.com/components/animated-modal>

**shadcn ui**

<https://ui.shadcn.com/>

**FakerJS :: is a popular library that generates fake (but reasonable) data**

<https://fakerjs.dev/guide/>

urlPicsumPhotos

Generates a random image url provided via <https://picsum.photos>.

<https://fakerjs.dev/api/image.html>

**NextUI is a UI library for React**

<https://nextui.org/docs/guide/introduction>

Route basics:

<Link href={“/products”}>Products</Link>

**DummyJSON**

Need dummy JSON data for your frontend? Look no further! DummyJSON Server serves up JSON data in a snap, eliminating backend setup headaches for rapid prototyping and testing.

<https://dummyjson.com/docs>

**{JSON} Placeholder**

**Free fake and reliable API for testing and prototyping.**

<https://jsonplaceholder.typicode.com/>

## Simplifying Data Fetching: How SWR Outperforms useEffect

<https://swr.vercel.app/docs/getting-started>

<https://www.getfishtank.com/blog/navigating-useswr-in-nextjs-applications>

**Comparing Schema Validation Libraries: AJV, Joi, Yup, and Zod**

<https://www.bitovi.com/blog/comparing-schema-validation-libraries-ajv-joi-yup-and-zod>

React Context

React Context is a way to manage state globally.

<https://www.w3schools.com/react/react_usecontext.asp>

**bcrypt.js (**To hash a password:**)**

npm i bcryptjs

<https://www.npmjs.com/package/bcryptjs>

Password = ‘sammy’

Hash = £%$^&£!23!3%!!

Salt = 2vqw£4Df$%sdfk

Hash + Salt = £%$^&£!23!3%!!2vqw£4Df$%sdfk

<https://www.digitalocean.com/community/tutorials/how-to-handle-passwords-safely-with-bcryptsjs-in-javascript>

**jsonwebtoken**

npm i jsonwebtoken

Here are some scenarios where JSON Web Tokens are useful:

Authorization:

**Information Exchange:**

<https://www.npmjs.com/package/jsonwebtoken>

<https://jwt.io/introduction>

[**Redux Toolkit**](https://redux-toolkit.js.org/)

The first thing to ask is, "what is Redux?"

Redux is really: <https://redux-toolkit.js.org/introduction/why-rtk-is-redux-today>

A single store containing "global" state

Dispatching plain object actions to the store when something happens in the app

Pure reducer functions looking at those actions and returning immutably updated state

// redux toolkit <https://redux-toolkit.js.org/> doc: <https://redux-toolkit.js.org/introduction/getting-started>

yarn add @reduxjs/toolkit

If you need React bindings:

yarn add react-redux

Follow the steps : <https://redux-toolkit.js.org/tutorials/quick-start>

1. *{type: 'cart/addToCart', payload: {…}}*
   1. **payload**: {id: 2, title: 'Eyeshadow Palette with Mirror', description: "The Eyeshadow Palette with Mirror offers a versati…it's convenient for on-the-go makeup application.", category: 'beauty', price: 19.99, …}
   2. **type**: "cart/addToCart"

here getting state object.

Payload is current item.

authjs

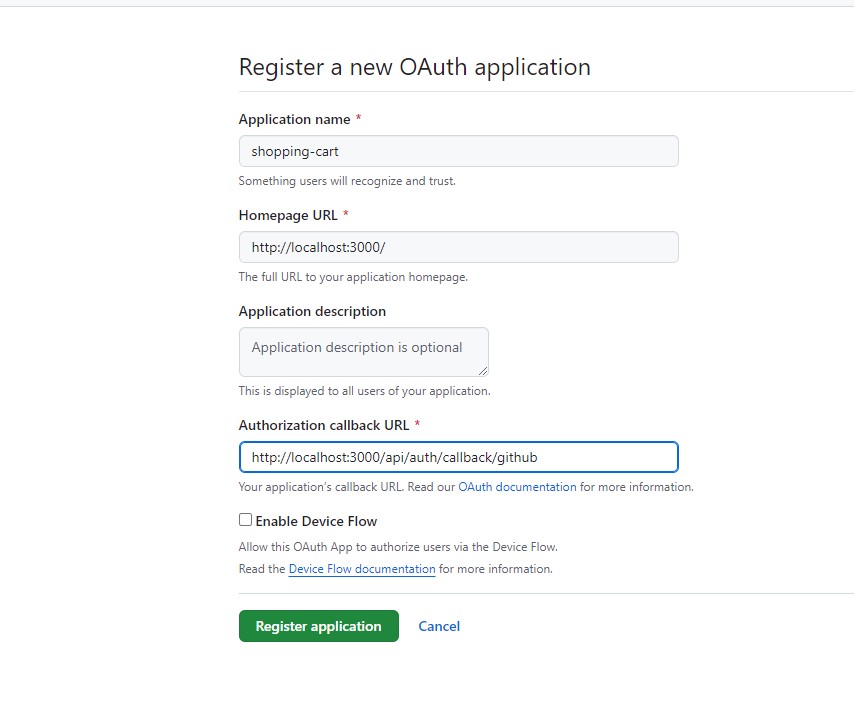
<https://authjs.dev/> doc install: <https://authjs.dev/getting-started/installation?framework=Next.js>

OAuth

### Register OAuth App in GitHub's dashboard

Go to github.com then Settings > Developer Settings

OAuth Apps // Register a new OAuth application



Then register application

Then npx auth secret

Create .env file and add

AUTH\_SECRET=zDsldfslkklkjsdkfjskdfjLKDJDKLKLEJHUDNskledfjksKJDJ

AUTH\_GITHUB\_ID=Ov23lie1ZNmjuq4ZTmEq

AUTH\_GITHUB\_SECRET=a40f89b42b457bbafbb16d7da3de07551031a4fd

In Next.js we recommend setting up your configuration in a file in the root of your repository, like at auth.ts.

Add the handlers which NextAuth returns to your api/auth/[...nextauth]/route.ts file so that Auth.js can run on any incoming request

import { handlers } from "@/auth"

export const { GET, POST } = handlers

Add optional Middleware to keep the session alive, this will update the session expiry every time its called.

export { auth as middleware } from "@/auth"