\*\*\*\*\*\*\*\*\*\*

Giới thiệu NextJS:

+ NextJS hosting trên công ty mẹ là Vercel (công ty cung cấp dịch vụ điện toán đám mây).

+ NextJS là framework của Vercel được build dựa trên React (của Meta).

+ NextJS vừa có thể là server vừa có thể là client => tuy nhiên nên dùng làm FE thôi.

+ Mặc định các component khi tạo ra trong NextJs là server component, khi muốn dùng các state, hook trong react phải khai báo thêm ‘use client’ ở đầu file.

NextJS giải quyết vấn đề gì?

+ Render website ở server nên thân thiện với SEO.

+ Khi bot của google vào website đọc nội dung => nế có sẵn HTML thì đọc nhanh hơn, còn nếu render ở client sẽ tốn nhiều thời gian => vượt qua thời gian chờ đợi => sẽ skip qua luôn.

+ Tích hợp nhiều tool mà ReactJS thuần không có.

+ Tối ưu image, font, script

+ CSS module

+ Routing

+ Middleware

+ Server Action

+ tích hợp SEO

+ Thống nhất cách viết code.

\*\*\*\*\*\*\*\*\*\*

Setup môi trường để code NextJS

+ Cần cài nodeJS, nên dùng npm (dễ dàng chuyển version)

+ Git (quản lý source code)

+ Tùy chọn:

+ Trình duyệt Chrome

+ IDE: Visual Studio Code

+ Setting visual code: <https://duthanhduoc.com/blog/cach-minh-setup-vs-code>

Cài đặt dự án NextJS

+ npx create-next-app@latest

+ Hỏi tên dự án: \_\_\_\_\_ tên folder sẽ tạo ra

+ Hỏi có muốn dùng typescript hay không? Có

+ Dùng ESLint hay không? Có (ESLint là bộ tool format chuẩn)

+ Hỏi dùng tailwind CSS không? Có

+ Hỏi có muốn tạo src trong dự án hay không? Có

+ Hỏi có muốn dùng App Router hay không? Có (nếu không sẽ chọn Page Router, nên dùng App Router).

+ Hỏi có muốn custom 1 cái import alias không? No (alias là kí hiệu để import một cách rút gọn hơn, nếu không thì sẽ chọn default là @, chọn có thì có thể chọn ký hiệu khác)

+ npm run dev: chạy next dev : chạy ở môi trường nhà phát triển

+ npm run build: chạy next build ; nén code lại cho nhỏ hơn

+ npm run start: nên chạy sau khi build

\*\*\*\*\*\*\*\*\*\*

Cấu trúc thư mục

+ Thư mục .next: tự tạo khi chạy npm run build, chưa code server khi build

+ public: chứa hình ảnh, video,…

+ src: chứa code chính

+ app: Đại diện cho App Router

+ page.tsx : tương đương index

+ layout.tsx : layout của file page.tsx

+ global.css: chưa code CSS

+ .eslintrc.json : cấu hình ESLINT, có thể thêm các rule để code

+ .gitignore : chặn các file không muốn up lên git

+ next-env.d.ts : định nghĩa các kiểu dữ liệu đặc biệt trong file này

+ next.config.mjs : chứa confid đặc biệt cho Next (ví đụ: config để hiển thị kích thức file build)

+ postcss.config.js: tailwind

+ tailwind.config.ts : config của tailwind, giám sát các file



+ tsconfig.json :



\*\*\*\*\*\*\*\*\*\*

Hướng dẫn sử dụng thư viện ShadCn UI

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

+ Hiện nay thứ viện UI thường chia làm 2 loại:

+ Style sẵn (MUI, Ant Design,…): phù hợp dự án đã có design sẵn theo phong cách cảu thư viện, hoặc dự án không cần design, code sao cũng được. Nhược điểm custom lâu và khó.

+ Unstyled (Radix UI, Prime React,…): Phù hợp mọi dự án, đặc biệt những dự án có design cá biệt hoặc custom cá biệt. Nhược điểm của thư viện này là hơi tốn thời gian custom.

+ ShadCn UI: bộ công cụ dựa trên tailwind

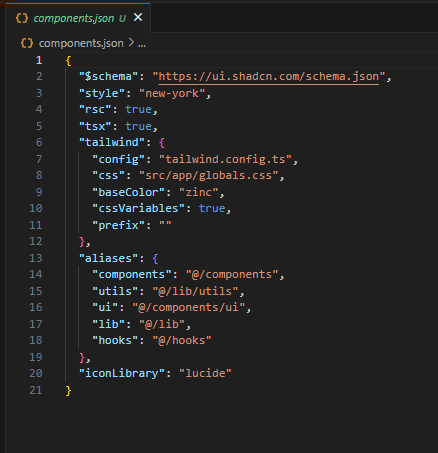
+ npx shadcn@latest init

+ Hỏi phong cách default/New York : New York chữ nhỏ hơn

+ Hỏi chọn màu ưa thích

+ Hỏi chọn CSS variable? Nên Yes

+ Khi tạo xong sẽ có components.json



+ Có thể cài bằng tay hoặc dùng shacn UI CLI

+ lib/utils.ts : chứa hàm tiện ích, giúp merge các class tailwind

+ vd: có class w-10 w-20, thì hàm sẽ xóa w-10 do overight

+ vd: Cài thử nút button : npx shadcn@latest add button

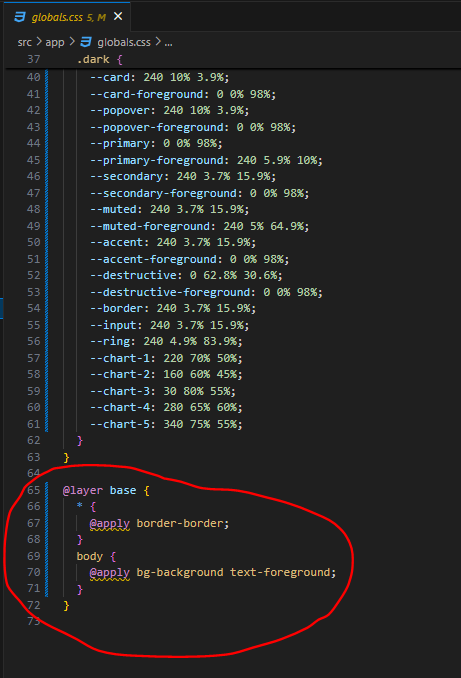
+ sẽ thêm tệp button.tsx vào thư mục components/ui => có thể import để tái sử dụng từ đây, có thể style lại bằng className, thì hàm cn sẽ merge lại

\*\*\*\*\*\*\*\*\*\*

CSS style trong NextJS

+ CSS Global:

+ Vào file global.css, file này đc import vào layout.tsx



+ Nhừng thành phần cơ bản nên đưa vào layer base, ví dụ như thẻ h1, h2,…

Khi đưa vào layer base sẽ tự sắp xếp theo thứ tự ưu tiên, các thẻ dưới sẽ overight các thẻ trên.

+ Tiện ích, ví dụ như font-size thì nên đưa vào layer utilities.

+ Có thể sử dụng custom.module.css để custom lại các class

+ Có thể dùng SASS bằng npm i sass

+ Dùng thư viện clsx (thường dùng tạo các toggle):

vd: className={clsx(‘card’ , {

[custom.card] : expanding //dùng state expanding true false để chạy

})}

\*\*\*\*\*\*\*\*\*\*

Sử dụng Font trong NextJS (Google, Local Fonts, Tailwind CSS)

+ Phải tuân theo rule của NextJS

+ Có 2 cách dùng thông thường:

+ Gọi từ CDN: code dễ, load lâu vì request đến CDN lâu

+ Hoặc host trên server: load nhanh vì load trang web sẽ load luôn font => sẽ ít xảy ra hiện tượng layout shift, do dùng font mặc định khi chưa kịp load font. (hostping font và caching font)

+ Dùng font từ google CDN:

+ Vào <https://fonts.google.com/>

+ Chọn ngôn ngữ ở tab ngôn ngữ : vì 1 số font ko hỗ trỡ TV nên phải test trước

+ Chọn Get Font

+ Get embed code (có thể chỉnh thêm tùy chọn ở tab bên trái) => nhúng link vào thẻ <head> html trong file layout.tsx => copy các class vào untilities của global.css

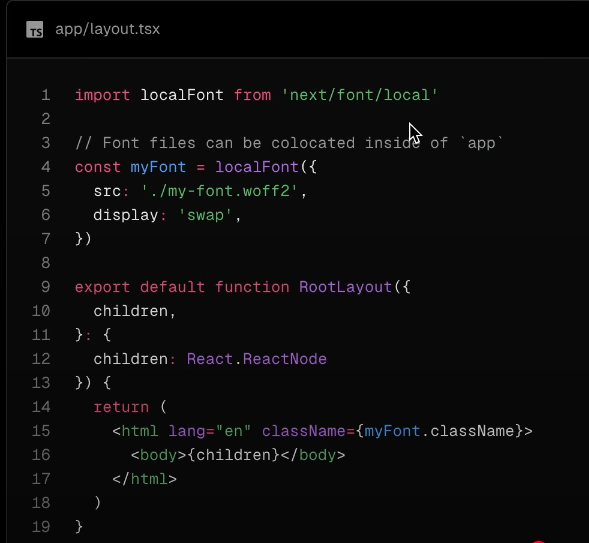
+ Có thẻ dùng các font trong ‘next/font/google’ của NextJS

+ được lưu trong .next khi build

+ Dùng font tải về:

+ Tải font cần dùng về

+ Dùng ‘next/font/local’



+ Kết hợp font với tailwind: dùng variable

\*\*\*\*\*\*\*\*\*\*

Sử dụng Image trong NextJS

+ Có 2 loại:

+ Lưu trong source code: thường lưu trong public, import Image form “next/image”

+ vd: <Image src=”/” alt=”alt” width={500} height={500} quality={100}/>

+ NextJs đã giảm đi dung lượng ảnh (do đó giảm chất lượng ảnh), có thể set bằng quality, mặc định quality 75, có thể set lên 100.

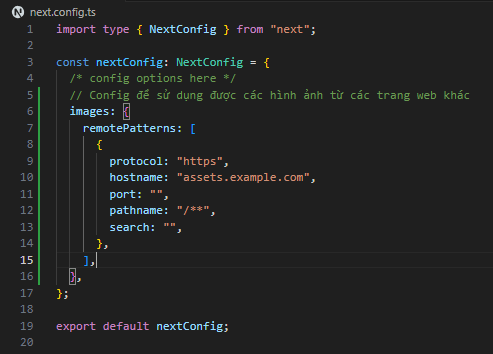
+ Có thể dùng className để style kích thước ảnh.

+ Nếu dùng image vẫn được nhưng ESLint sẽ báo lỗi bị low LCP chỉ số đánh giá website

+ Nhúng từ link bên ngoài:

+ NextJs sẽ chặn để tránh bị tấn công bên ngoài

+ Config lại ở file



\*\*\*\*\*\*\*\*\*\*

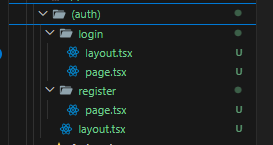
Routing trong NextJs

+ https://nextjs.org/docs/app/building-your-application/routing

+ Routing được quy định dựa trên tên folder

+ Cần có file page.tsx ứng cho từng trang của folder

+ Nếu nhóm 2 folder lại thì phải có dấu (), tham khảo mục route group.



+ Có thể tạo layout con

+ Chuyển trang có thể tham khảo mục Linking and Navigating

+ Dùng thẻ Link, import Link from “next/link” :

+ Dùng hook useRouter() , dùng từ “next/navigation” cho App Router

+ Để dùng được hook phải thêm “use client”

+ Cách để sử dụng hook này, cũng có thể khai báo 1 component riêng để nó là client component, nên giữ page là server component nhiều nhất có thể

* Nên dùng thể Link vì nó tốt hơn về mặt UX

+ Dùng redirect, import {redirect} from “next/navigation” : chạy ở server component, có thể gọi trong client component, nhưng không có thể chạy trong event handler.

\*\*\*\*\*\*\*\*\*\*

Khái niệm về Rendering

+ Rendering là convert code thành giao diện người dùng có thể tương tác được

+ NextJS và ReactJS cho phép render ở server và client, gọi là Hibrid Application

+ Client: đại diện trình duyệt người dùng

+ Server: đại diện cho máy chủ nơi chứ data và trả response về

+ Client và Server là 2 môi trường tách biệt nhau. Đây gọi là “Network Boudary”, ranh giới mạng.

+ Muốn code chạy ở client phải dùng “use client”.

+ Client lấy data ở client chỉ cần truy cập vào object ở client, Client muốn lấy data ở server phải gửi 1 request đến server.

\*\*\*\*\*\*\*\*\*\*

Client Component trong NextJS

+ Single Page Application (SPA) – React SPA truyền thống (React Vite, CRA, …) là 1 client component khổng lồ.

+ Khi lần đầu vào trang web:

+ Trình duyệt request đến server và trả về file index.html cơ bản ( hầu như không chứa html gì nhiều).

+ Trình duyệt nhận thấy trong file html có link đến file js, css nên request lần nữa đến server để lấy file js, css.

+ Trình duyệt tiến hành chạy code JS để render ra HTML và gắn sự kiện vào HTML

+ Người dùng thấy và tương tác được với tang web

+ Trong quá trình này, web sẽ trắng xóa cho đến khi bước thứ 3 được hoàn thành.

+ Vậy nên mới nói lần đầu tiên khi truy cập vào các SPA truyền thống khá lâu, nhưng sau đó thì chuyển trang rất nhanh vì js bundle cả app đã có ở client rồi, nếu cần data mới request đến server.

+ Tại bước 3, lúc nào HTML cũng được JS trình duyệt render ra khi chúng ta truy cập vào web. Cái này gọi là Dynamic Rendering.

+ Dùng client component trong NextJS:

+ Cần tương tác: dùng hook, useState, useEffect, event listener (onClick, onSubmit, onChange,…),…

+ Cần dùng các API từ trình duyệt. vd. (window.href,..)

+ Trong NextJS tất cả các component đều được render ra HTML sẵn khi có thể lúc NextJS build (Static rendering). Kể cả server component hay client component.

+ Do đó, khi truy cập vào 1 trang web NextJS, sẽ thấy UI ngay lập tức do server NextJS trả về HTML đã render sẵn. Sau đó trình duyệt sẽ render lại client component 1 lần nữa để đồng bộ DOM, sự kiện, state, effect.

+ Nhận thấy:

+ Client component bị render tối thiểu 2 lần: 1 lần khi build, 1 lần trình duyệt render lại.

+ Trả về HTML sẵn nên thấy content ngay lập tức (tăng UX).

+ Dù thấy content ngay lập tức nhưng vẫn không thể tương tác ngay được vì cần phải chờ trình duyệt đồng bộ lại client component (render, gắn sự kiện, state, effect,…)

+ Ưu điểm của Client Component:

+ Giảm gánh năng cho server khi component nặng và phức tạp về logic => Server yếu thì nên dùng .

+ Nhược điểm:

+ SEO không tốt

+ Thiết bị client yếu thì chạy không nổi.

+ Tăng bundle size javascript

* Dùng server component khi có thể. Không đặt năng vấn đề cấu hình server, vì dùng cho production thì server phải tốt. Quan trọng là trải nghiệm người dùng.

\*\*\*\*\*\*\*\*\*\*

Server Component

+ Đây là chế độ mặc định của component trong NextJS

+ Ưu điểm:

+ Fetch data ở server => Nơi gần data center nên là sẽ nhanh hơn là fetch ở client => Giảm thiểu thời gian rendering, tằng UX.

+ Bảo mật: Server cho phép giữ các data nhạy cảm, logic đặc biệt không muốn public ở client.

+ Caching: vì được render ở server nên có thể lưu giữ cache cho nhiều người dùng khác nhau => Không cần render trên mỗi request

+ Bundle size: Giảm thiểu JS bundle size vì client không cần tải về phần JS logic trên render HTML.

+ Load trang lần đầu nhanh và chỉ số FCP (First Contentful Paint) thấp do người dùng sẽ thấy content ngay lập tức.

+ Search Engine Optimization and Social Network Shareability (khả năng share trên mạng xã hội): SEO sẽ không đợi website render, nếu quá thời gian nó skip qua luôn.

+ Streaming

\*\*\*\*\*\*\*\*\*\*

Cách NextJS render component

+ Mọi component dù là server component hay là client component khi build đều sẽ có:

+ Static HTML

+ JS bundle

+ Ngoài ra còn có CSS bundle, Image, Font,…

+ Khi request lần đầu tiên đến trang web NextJS:

+ Server NextJS render server component và kết hợp với Client component để tạo ra HTML để gửi về client.

+ Client ngay lập tức thấy được website nhưng chưa tương tác được. (ví dụ Click, hover,..)

+ Trong JS Bundle download về có chưa React Server Component Payload (RSC Payload), cái này dùng để render lại client component ở client, cập nhật DOM.

+ Cuối cùng là sẽ thêm các sự kiện vào các client component để tương tác với người dùng => bước này gọi là Hydration, sau bước này thì có thể tương tác với website.

* React Server Component Payload là 1 data đặc biệt được render ở phía server phục vụ cho việc đồng bộ, cập nhật DOM giữa các Client Component và Server Component.

+ Khi request lần thứ 2 (Subsequent Navigation)

+ ví dụ: navidate từ /home sang /about

+ Thì server NextJS sẽ không trả về HTML nữa, mà trả React Server Component Payload (RSC Payload) và các bundle JS, CSS cần thiết.

+ Client sẽ tự render ra HTML

+ Điểu này sẽ giúp navigation nhanh hơn, nhưng vẫn đảm bảo về SEO.