Chapter 0: Giới Thiệu Về Khóa Học **2**
#0.1. Demo Kết Quả Đạt Được Khi Kết Thúc Khóa Học 2
#0.2. Về Tác Giả 2
Chapter 1: Setup Environment **3**
#1. NextJS là gì ? 3
#2. Setup Môi trường 5
#3. Yêu cầu trước khi bắt đầu 6
#4. Cách Học Khóa Học Hiệu Quả 7
#5. Lịch sử version của Next.JS 8
Chapter 2: Hello world với Next.JS **9**
#6. Hello world 9
#7. Cấu trúc dự án Next.js cài đặt 10
#8. Lưu ý về lựa chọn tài liệu 11
#9. Client Component (SPA) 12
#10. Server Component 14
#11. Tại sao cần Next.js ? 15
#12. Tư duy sử dụng component với Next.js 16
Chapter 3: Router & Layout **17**
#13. Giới thiệu về Routing 17
#14. Định nghĩa Route (Page) 18
#15. Bài tập tạo Page 19
#16. Điều hướng trang Navigation 19
#17. Layout 20
Chapter 4: Practises **21**
#18. Sử Dụng CSS với Next.JS 21
#19. Tích Hợp React Bootstrap 22
#20. Bài tập tạo base giao diện Home Page 23
#21. Setup Dự Án Backend 24
#22. Fetching data với Next.js 25
#23. useSWR 26
#24. Display List Blog 27
#25. Design Modal Add New 28
#26. API Add New 29
#27. Mutate Data với SWR 30
#28. Bài tập Update Blog (modal + api) 31
#29. Tạo Page by id 32
#30. Bài Tập Design Page view detail 33
#31. API View detail 34
#32. Bài tập delete 35
#33. Meta data 36
#34. Tổng kết 37

Chapter 0: Giới Thiệu Về Khóa Học

#0.1. Demo Kết Quả Đạt Được Khi Kết Thúc Khóa Học

Video demo: https://youtu.be/QNd76WGGCZs

Một vài highlight của khóa học:

- **Hiểu rõ, thực hành và áp dụng được ngôn ngữ Typescript cho dự án Next.js**
- **Xây dựng tư duy frontend với Next.JS và React**

#0.2. Về Tác Giả

Mọi thông tin về Tác giả Hỏi Dân IT, các bạn có thể tìm kiếm tại đây:

Website chính thức: https://hoidanit.com.vn/

Youtube "Hỏi Dân IT" : https://www.youtube.com/@hoidanit

Tiktok "Hỏi Dân IT" : https://www.tiktok.com/@hoidanit

Fanpage "Hỏi Dân IT" : https://www.facebook.com/askITwithERIC/

Udemy Hỏi Dân IT: https://www.udemy.com/user/eric-7039/

Chapter 1: Setup Environment

#1. NextJS là gì ?

Lưu ý: đây là quan điểm cá nhân của mình, ae thấy điều gì chưa đúng, cần bổ sung cứ comment nhé.

What ?

- Nextjs là open-source project, tạo bởi Vercel

- Next.js là fullstack website, cung cấp giải pháp render dữ liệu sử dụng React với Server (SSR/CSR với React)

- Là framework cho react, recommend bởi react team: https://react.dev/learn/start-a-new-react-project#nextjs

Why ?

- **React là thư viện, không phải là framework:**

+ ưu điểm: **flexible**, bạn có thể tổ chức code theo ý muốn. tốt/xấu do bạn quyết định (all cases)

+ nhược điểm: không có giải pháp thống nhất (kiến trúc) cho : structure (cách tổ chức code), **router** (điều hướng trang), **manage state** (quản lý state)...

=> Next.js ra đời để giải quyết vấn đề trên

- Ngoài ra, Nextjs được tối ưu để tốt cho SEO **(Search Engine Optimization**)

2. Next.JS là frontend hay backend ?

It's depend on your needs. Nếu framework giải quyết tốt bài toán bạn gặp phải => ok

Nextjs chỉ nên dùng làm frontend:

- Cấu trúc tổ chức hiện tại của Next hỗ trợ rất tốt cho frontend, nếu thêm backend vào => thêm thành phần, khó maintain

=> công việc của backend có thể dùng framework như Nest.js

- Next.js có hỗ trợ backend, tuy nhiên chỉ dừng ở mức basic (forward request tới server đích)

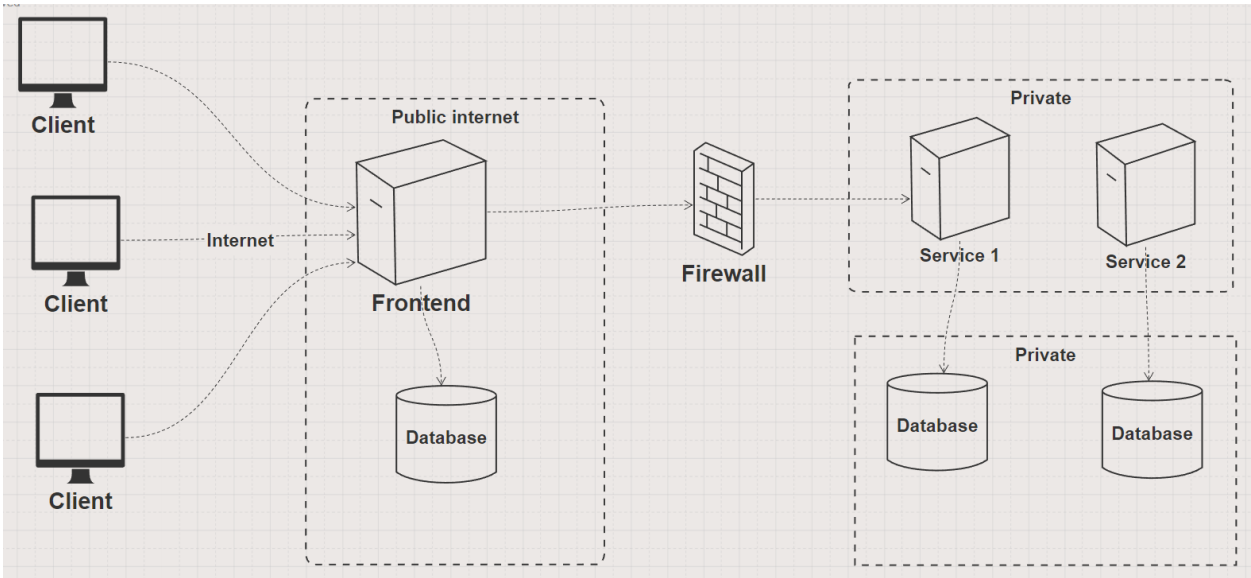
+ giảm thiểu khối lượng code cho Next.js

+ lấy được ưu điểm render dữ liệu từ server trước khi trả về cho client

+ Đôi khi cần dấu thông tin như API keys => dùng server render (perfect :v)

- Vấn đề bảo mật, nếu Next.js làm nhiệm vụ backend => phơi trực tiếp ra internet (anyone can access)

Bonus: mô hình tổ chức nếu Next.JS làm frontend



#2. Setup Môi trường

1. Công cụ code (IDE)

Lựa chọn vscode vì hỗ trợ mạnh mẽ javascript/typescript

Download vscode: https://code.visualstudio.com/download

+ setup format on save

+ only use extensions:
Auto Close Tag , Auto Complete Tag, Auto Rename Tag => code HTML
Code Spell Checker => check lỗi chính tả khi đặt tên biến = tiếng anh

2. Môi trường code

- **Sử dụng Node.JS v18.16.0**
Download node.js v18: https://nodejs.org/download/release/v18.16.0/

Nếu muốn sử dụng nhiều version của node.js, sử dụng nvm:
https://www.youtube.com/watch?v=ccjKHLyo4lM&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9VqI&index=40

3. Git

Máy tính đã cài đặt Git : **git -v**

Học nhanh về Git tại đây:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo

#3. Yêu cầu trước khi bắt đầu

1. Có hiểu biết cơ bản về HTML/CSS
- Biết CSS/SCSS (modules)

2. Có hiểu biết cơ bản về JavaScript/Typescript
Ở đây là hiểu biết về cú pháp và cách khai báo sử dụng biến/functions

- Xem nhanh về Javascript:
https://www.w3schools.com/js/
https://www.youtube.com/playlist?list=PLncHg6Kn2JT5dfQqpVtfNYvv3EBVHHVKo

- Xem nhanh về Typescript:
https://www.youtube.com/playlist?list=PLncHg6Kn2JT5emvXmG6kgeGkrQjRqxsb4

3. Có hiểu biết cơ bản về React (state/props/components)

- Xem nhanh về React Hook :
https://www.youtube.com/playlist?list=PLncHg6Kn2JT45Gg8eOGiqHv2yFqcBXJrK

4. Biết cách sử dụng Git
- Xem nhanh về Git tại đây: (git pull, git push , git commit…)
https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo

#4. Cách Học Khóa Học Hiệu Quả

Tất cả công nghệ sử dụng đều là mới nhất tại thời điểm mình quay video :v

1. Công nghệ
- React 18
- Next 13
- Typescript

2. Cách học
- Học theo tài liệu của Nextjs (**app folder** structure)

3. Hỗ trợ giải đáp
- **comment youtube** (lưu ý đọc comment của người khác)

- **group community** (lưu ý ko inbox) : https://www.facebook.com/groups/hoidanit

#5. Lịch sử version của Next.JS

4/2022: React 18 ra đời, hỗ trợ mạnh mẽ render React ở Server (server component)
5/2023: Next 13 ra đời (hết giai đoạn beta), hỗ trợ các tính năng của React 18.

Và có nhiều breaking change (so với các version cũ < v.13)

Hướng Dẫn Đọc Document Version Cũ của Next.JS (dùng cho Next <= 12)

Link github của Next.js: https://github.com/vercel/next.js

1. chọn thư mục docs => chọn tags
ví dụ:https://github.com/vercel/next.js/tree/v12.3.4/docs

https://github.com/vercel/next.js/tree/v13.4.2/docs

2. Download thư mục trên

https://download-directory.github.io/

=> paste url vào, ok :v

3. Chạy với docusaurus : convert markdown => html

https://docusaurus.io/

Các bước cần làm: (cần môi trường Node.JS)
- create new project: npx create-docusaurus@latest my-website classic

xóa hết files trong thư mục docs
copy paste md files => paste vào thư mục docs

- chạy npm run start => check lỗi

=> remove file upgrading.md

Chapter 2: Hello world với Next.JS
#6. Hello world
warning: Tất cả đường link docs tham khảo có thể bị break trong tương lai. lưu ý xem #5 để biết cách đọc tài liệu quá khứ (dựa theo version)

Tài liệu: https://nextjs.org/docs/getting-started/installation

Lưu ý:
- về môi trường cài đặt: Node.js version (v18.16.0)
- thực hiện clone code (bước 1) để đảm bảo code được sử dụng là giống nhau

1. Clone code
Link github: https://github.com/harypham/next13-starter-typescript

Các bước cần làm:
- clone source code: git clone ...
- cài đặt thư viện
- run project: npm run dev

2. Run with command
Chỉ thực hiện bước này sau khi bạn đã hiểu Next.js là gì, và đã có khả năng tự fix bugs

sử dụng câu lệnh: **npx create-next-app@latest**

#7. Cấu trúc dự án Next.js cài đặt

Thư mục:
.next: đây là code "đã được dịch" của Next.js (cái mà browser chạy)

- **public**: chứa static file (CSS/JS/images)
- **src/app** : chứa source dự án với Next 13 (router)

.**eslintrc.json** : config eslint (warning ở terminal :v)

.gitignore: các files ko push lên git remote

next.config.js : cấu hình Next.js

package.json : thông tin thư viện cài đặt

tsconfig.json : cấu hình compiler của typescript

#8. Lưu ý về lựa chọn tài liệu

Đây là series về Next.JS version 13, có rất nhiều thay đổi so với Next 12 và các version trước đó.

Next13 “đập đi” rất nhiều cái đã có, chia ra thành khái niệm “server” và “client”.

Vì vậy, khi học Next 13, hãy quên đi những điều bạn đã biết về Next.js trước đây

#9. Client Component (SPA)
React, Angular, Vue là những thư viện/framework chuyên làm UI cho client.
Client component chính là CSR (client side rendering)

1. What ?
Ví dụ: https://hoidanit.com.vn/

Client component ám chỉ quá trình render giao diện (component) xảy ra tại client (hay còn biết tới là SPA - single page application)

Quy trình thực hiện:

1. Client vào trang web, đây là quá trình khởi tạo request đầu tiên lên server

2. Server nhận yêu cầu, trả về duy nhất file index.html

3. Client nhận file index.html để render dữ liệu cho người dùng. Kể từ đây, việc 'tương tác giao diện' của người dùng
sẽ được xử hoàn toàn 100% ở browser của client.

4. Trong trường hợp client cần hiển thị thông tin động (dynamic), ví dụ như hiển thị table, cần khởi tạo request lên server để lấy dữ liệu.

Ở đây, sử dụng hình thức Restful APIs, như vậy vẫn đảm bảo được trải nghiệm của client mà không cần reload website

2. Ưu điểm
- Tăng trải nghiệm của người dùng (SPA - single page application), dễ thấy nhất là "ít khi" cần reload lại website. Ví dụ Facebook

- Giảm thiểu "gánh nặng cho server", việc tính toán và xử lý giao diện sẽ nằm 100% ở client

- Server với cấu hình yếu vẫn hoạt động ok (vì có cần làm gì đâu :v)

3. Nhược điểm

- Nếu browser (máy tính, điện thoại...) của client là thiết bị yếu (RAM, CPU..., hoặc slow network), quá trình render "lần đầu" sẽ tốn thời gian.

Minh họa = chế độ slow network với website https://hoidanit.com.vn/

Việc khởi tạo request đầu tiên lên server (lấy file index.html) về để render giao diện, vì bây giờ, tất cả "giao diện của website" đều phải được load trước.

Nó là đánh đổi. muốn "server nhàn" thì "client sẽ bận" thôi mà :v

#10. Server Component
Minh họa: dự án node.js tự code

What ?

Server Component, có thể hiểu là SSR (server side rendering), là cách render dữ liệu 100% ở phía server.

Quy trình:

Client truy cập route "/" => server render dữ liệu tương ứng với route "/" => gửi kết quả cho client

Client truy cập route "/about" => server render dữ liệu tương ứng với route "/about" => gửi kết quả cho client

Ưu điểm:
- Client chỉ "khởi tạo request" và "nhận kết quả" => tốc độ load trang web rất nhanh

Nhược điểm:
- Việc tối ưu hóa UI trên server khá khó code, tốn time và khó maintain (code chay HTML, CSS kết hợp với template engine)

- Cần cấu hình server đủ mạnh (RAM, CPU) => tốn money :v

#11. Tại sao cần Next.js ?

Vấn đề đặt ra:

- **Nếu dùng CSR (client side rendering),** tốc độ load trang sẽ chậm (nếu thiết bị/network của client yếu)

Đổi lại, code UI "rất sướng" khi sử dụng React/Angular/Vue, đồng thời, chi phí duy trì server nó "rẻ"

- **Nếu dùng SSR (server side rendering), t**ốc độ load trang nhanh hơn nhiều so với CSR (kể cả thiết bị/network của client yếu)

Đổi lại, code UI "khổ" vì cần code chay HTML, CSS (với template engine), đồng thời, chi phí server nó "cao hơn CSR"

Next.JS sinh ra để giải quyết vấn đề trên. (hybrid app)

- next.js là server (backend), nó đảm bảo được tốc độ load trang nhanh

- để khắc phục việc code UI "khổ" tại server, Next.js hỗ trợ code "React" ở server => amazing

- Cần 1 server ở cấu hình trung bình là đã chạy được Next.js. wow :v

=> Như vậy, Nextjs là giải pháp để dung hòa giữa việc sử dụng chỉ mình "CSR" hay "SSR" như cách truyền thống

#12. Tư duy sử dụng component với Next.js

Minh họa tiki book => sử dụng zoom layout + slow 3G

Tài liệu:
https://nextjs.org/docs/getting-started/react-essentials

Next.js cho phép sử dụng SSR (render dữ liệu ở server => gửi về client) =>
gọi là server component

hoặc CSR (render dữ liệu ở client) => gọi là client component

1. Khi nào nên sử dụng server component/client component ?

Tham khảo:
https://nextjs.org/docs/getting-started/react-essentials#when-to-use-server-and-client-components

- Đối với dữ liệu "ít thay đổi": logo, category... => dùng server component (render trước ở server)

- Đối với dữ liệu thay đổi "thường xuyên, phụ thuộc vào tương tác của người dùng" => client component

2. Tại sao lại chia tắt Server/Client Component ?
- Với dữ liệu "tĩnh" => server render
- Với dữ liệu "động" => client render

Việc chia tách dữ liệu "tĩnh + động" => sẽ làm giảm kích thích file cần render ở client (vì đã render 1 phần ở server)

=> tốc độ load trang web sẽ nhanh hơn.

Chapter 3: Router & Layout
#13. Giới thiệu về Routing
Tài liệu: https://nextjs.org/docs/app/building-your-application/routing

- Với Next 13, chúng ta sử dụng thư mục "app" để khai báo route.

- Mặc định, **tất cả "component" khai báo bên trong thư mục "app" sẽ là SERVER COMPONENT**

- Với Next < 13 (v12 trở về trước). chúng ta có thư mục "pages" để khai báo route Component khai báo bên trong thư mục "pages" là client component

Khi học Next 13, không cần thư mục "pages" nữa, vì Next sẽ cung cấp giải pháp mới giúp phân biệt "client component" và "server component".

Nếu như bạn join 1 dự án có cả "pages" và "app" folder
=> đây là dự án migrate lên next 13.

=> tất cả code mới, nên viết bên trong thư mục "app" để tận dụng tối đa sức mạnh của Next 13.

#14. Định nghĩa Route (Page)
Tài liệu: https://nextjs.org/docs/app/building-your-application/routing/defining-routes

Nếu bạn có background là React developer:
1. Nguyên tắc cần tuân theo:
File-system based routing
- Do NextJS là framework, bạn không cần phải tích hợp thư viện "react-router" như cách truyền thống, Nextjs đã tích hợp sẵn cho bạn

- Không cần phải "tự định nghĩa route + component",
ví dụ : <Route path="teams" element={<Teams />}>

thay vào đấy là định nghĩa route dựa vào cấu trúc thư mục của Next.js

- Tất cả route trong ứng dụng, đều được định nghĩa bên trong thư mục "app"

- Tên thư mục viết thường
(không dùng các viết có chữ hoa hay thường, ví dụ folderName)

2. Các bước thực hiện:

route "/" => component render là "page.tsx" nằm ở ngoài cùng

route "/admin" => cần tạo folder "admin" và **tạo file page.tsx**

route "/admin/dashboard" => cần tạo folder "admin" => folder "dashboard" và tạo file page.tsx ở folder "dashboard"

Ý tưởng ở đây là, **mỗi folder, là 1 thành phần trên đường link URL**. file page.tsx chính là giao diện component được render ứng với route đấy.

p/s: do dùng typescript, nên khai báo là page.tsx. nếu bạn dùng javascript => khai báo là page.js, page.jsx,

#15. Bài tập tạo Page

- Tạo home page: "/" display 3 options:

- Tạo '/youtube' => hiển thị thông tin

- Tạo '/facebook' => hiển thị thông tin

- Tạo '/tiktok' => hiển thị thông tin

#16. Điều hướng trang Navigation
Tài liệu:
https://nextjs.org/docs/app/building-your-application/routing/linking-and-navigating

#17. Layout
Tài liệu:
https://nextjs.org/docs/app/building-your-application/routing/pages-and-layouts#layouts

Nguyên tắc:
- Layout giúp bạn reuse-code (share phần chung html cho các component khác nhau)

- Cần ít nhất/tối thiểu 01 layout để Next.js hoạt động

- file layout là React component, có tên bắt buộc là "layout.tsx" (dùng cho typescript)

- Có 2 loại layout: root layout và layout bạn tự định nghĩa (dùng cho route bạn muốn).

1. Root layout
- là file layout.tsx bên trong thư mục "app"
- cần định nghĩa tags: <html> và <body>

- Root layout là server component => không thể là server component

2. Nested layout
- khi có nhiều layout.tsx => layout sẽ tự động được "nested"

Chapter 4: Practises

#18. Sử Dụng CSS với Next.JS

Tài liệu: https://nextjs.org/docs/app/building-your-application/styling

2 trường phái code CSS:

1. Code CSS thuần (tương tự như HTML), gồm có
   - CSS
   - SASS
   - Tailwind (utility-first CSS framework)

2.CSS in JS (styled component - tương tự React Native)

#19. Tích Hợp React Bootstrap

Cài đặt thư viện:

npm i --save-exact react-bootstrap@2.8.0 bootstrap@5.3.0

Upgrade next => 13.4.12

#20. Bài tập tạo base giao diện Home Page

- Header
  https://react-bootstrap.netlify.app/docs/components/navbar#overview

- Footer

- Table hard code display block
  https://react-bootstrap.netlify.app/docs/components/table

https://react-bootstrap.netlify.app/docs/layout/grid/#container

#21. Setup Dự Án Backend

Link dự án backend: https://github.com/harypham/backend-fake-json

#22. Fetching data với Next.js
F12, check tab network
Fetch API:
https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Todo: Youtube/Tiktok => sử dụng Link component

#23. useSWR
Tài liệu: https://swr.vercel.app/

Cài đặt:
npm i --save-exact swr@2.2.0

Mục đích của SWR là cache data khi load web phía client

#24. Display List Blog

Todo://

#25. Design Modal Add New

https://react-bootstrap.netlify.app/docs/components/modal#static-backdrop

- Design base modal

- Design form with input

https://react-bootstrap.netlify.app/docs/forms/form-control

#26. API Add New

Tài liệu: https://www.npmjs.com/package/react-toastify

Cài đặt react toastify:

npm i --save-exact react-toastify@9.1.3

API Add New blog:

POST /blogs

body truyền lên object { title, author, content}

Sử dụng POST với fetch: https://stackoverflow.com/a/29823632

#27. Mutate Data với SWR

Tài liệu: https://swr.vercel.app/docs/mutation

#28. Bài tập Update Blog (modal + api)
Tạo modal Update tương tự như modal Create

API Update Blog:
PUT /blogs/id

Lưu ý: truyền động id vào url

#29. Tạo Page by id

- Tạo page blogs : /blogs

- Tạo page theo id: https://nextjs.org/docs/app/building-your-application/routing/dynamic-routes

#30. Bài Tập Design Page view detail

Tài liệu: https://react-bootstrap.netlify.app/docs/components/cards#header-and-footer

#31. API View detail

//todo

https://swr.vercel.app/docs/typescript

#32. Bài tập delete
API Delete Blog:
DELETE /blogs/id

Lưu ý: truyền động id vào url.

https://www.w3schools.com/jsref/met_win_confirm.asp

#33. Meta data

https://nextjs.org/docs/app/building-your-application/optimizing/metadata

#34. Tổng kết

Deploy với Vercel?

What's next ?