

TÀI LIỆU KHÓA HỌC “React Pro++ với Next.js”

Tác giả: Hồi Dân IT & Eric

Version: 3.0

Note cập nhật:

- Hướng dẫn upgrade Next.js 14
- Bổ sung kiến thức về Server Actions với Next.js 14
- Bổ sung chapter 21, 22, 23

Chapter 0: Giới thiệu	7
#0. Demo Kết quả đạt được	7
#1. Hướng Dẫn Download Tài liệu khóa học	11
#2. Yêu cầu của khóa học	12
#3. Về khóa học này	13
#4.1 Hướng Dẫn Sử Dụng Khóa Học Hiệu Quả	15
#4.2 Về tác giả	16
Chapter 1: Setup Environment	17
#5. Công cụ code IDE	17
#6. Browser	18
#7. Node.JS	20
#8. Quản lý code với Git	21
Chapter 2: Ôn tập Javascript	22
#9. Cách học kiến thức Javascript hiệu quả	22
#10. Destructuring Assignment	23
#11. Spread syntax	26
#12. Conditional Operator (Toán tử điều kiện)	28
#13. Optional Chaining	29
Chapter 3: React Basic	31
#14. Lưu ý về cách code React	31
#15. React là gì ? Phân loại dự án với React	32
#16. Setup Project React với Vite	35
#17. Cấu trúc dự án React Vite	36
#18. Think in React	37
#19. Component	38
#20. Import/Export Component	40
#21. React JSX	42
#22. Sử dụng Variables với JSX	43
#23. Bài tập design Component Input	44
#24. Props là gì	45
#25. Props với Typescript	46
#26. React Devtool	47
#27. Khái niệm Re-render	48
#28. Rendering Lists	49
#29. Key	50
Chapter 4: React Hook	51
#30. Event Handler	51
#31. State là gì ?	52
#32. useState Hook	53
#33. Bài tập về State (part 1)	54
#34. Bài tập về State (part 2)	55
#35. Passing Function từ Cha sang Con (Parent to Child)	56
#36. Lift up State	57

#37. Vòng đời của component	58
#38. Về React Cơ Bản	59
Chapter 5: Setup Dự Án Backend	61
#39. Giới thiệu tổng quan về dự án thực hành	61
#40. Cài đặt Postman	62
#41. Cài đặt Mongodb Compass tại local	63
#42. Setup database MongoDB Atlas	64
#43. Kích hoạt dự án backend	65
#44. Run dự án backend	66
Chapter 6: Routing với React (Client Side)	67
#45. React Router	67
#46. Tạo Route Users	68
#47. Design table User	69
#48. Backend và API là gì	70
#49. Postman để test API	71
#50. Gọi API với React	72
#51. Mô hình stateless	73
#52. Hiển thị list User	74
#53. Sử dụng key React hiệu quả	75
Chapter 7: Hướng dẫn sử dụng Chrome Dev Tool (Bồ Trợ)	78
#54. Thành phần của dev tool	78
#55. Kỹ thuật check api	79
#56. Nguyên tắc khi gọi API không có kết quả	80
Chapter 8: Module Users	81
#57. Tích hợp antd	81
#58. Design giao diện Admin	82
#59. CSS với React	83
#60. Tích hợp SASS (SCSS)	84
#61. Antd Table	85
#62. Hiển thị danh sách Users	86
#63. Antd Modal	86
#64. Design Modal Add New	87
#65. API Add New User	88
#66. Bài tập Add New User	89
#67. Bài tập Design Modal Update User	90
#68. Bài tập Update User	91
#69. Design Delete User	91
#70. Bài tập Delete User	92
#71.1 Pagination	93
#71.2 Antd Pagination	95
#71.3 Update Token	95
#72. Bài tập Paginate User	96
Chapter 9: Tối ưu hóa hiệu năng Form React	97
#73. Controlled Component vs Uncontrolled Component	97

#74. Thư viện hỗ trợ Uncontrolled Component	98
#75. Create User (Uncontrolled Component)	99
#76. Update User (Uncontrolled Component)	99
Chapter 10: React Hiện Đại với Next.JS	100
#77. Vấn đề tồn đọng với cách viết React cũ (CSR)	100
#78. Nextjs là gì	101
#79. Nextjs làm frontend hay backend ?	102
#80. Setup dự án với Nextjs (tích hợp MUI)	103
#81. Cấu trúc dự án Next.js	104
#82.1 React children	105
#82.2 React Fragment	105
#83. Cách đọc tài liệu của Nextjs	107
#84. Chuyên Next.js ngốn RAM	108
Chapter 11: Next Routing	109
#85. Tạo base giao diện	109
#86. Routing với nextjs	110
#87. Định nghĩa route	111
#88. Phân biệt cách code UI với React	112
#89. MUI là gì	115
#90. Cách sử dụng MUI component	116
Chapter 12: Module HomePage	117
#91. Bài tập Design App Bar	117
#92. Navigating	118
#93. Share Layout	118
#94. Server Component	119
#95. Client Component	120
#96. Use Client	121
#97. Bài Tập Design Main Content	122
#98. Design Footer	123
#99. Nextjs Environment Variables	124
#100. Fetch music from backend	125
#101. Fetch data với Nextjs	126
#102. Fetch Data HomePage	127
#103. Fetch Wrapper	128
#104. Hoàn thiện HomePage	130
Chapter 13: Module Wavesurfer	131
#105. Dynamic Route	131
#106. Search Params	132
#107. Giới thiệu về Wavesurfer	133
#108. Next.JS Public Folder	134
#109. Wavesurfer Basic	135
#110. React Ref	136
#111. Fetch track from Remote	137
#112. Tạo wavesurfer hook	140

#113. Sử dụng useMemo hook	141
#114. Handle Event với useCallback	142
#115. Add Bar Width	143
#116. HTML Canvas Gradients	144
#117. Add Gradients	144
#118. Add Time	145
#119. Add Hover	146
#120. Remove ID by Ref	146
#121. Add Customize Bar	147
#122. Add Comment	148
#123. Add Tooltip	149
Chapter 14: Module Auth	150
#124. Tài liệu về Authentication với Next.js	150
#125. Giới thiệu về Next-auth	151
#126. Test Next Auth	152
#127. Debugs với Next.js và VSCode	153
#128. Config Next-Auth Routes	155
#129. Config Github Provider	156
#130. Luồng hoạt động của Next-auth ?	157
#131. Config Session Cho Client	159
#132. Config Session cho Server	161
#133. Customize Session	162
#134. Login with Credentials	167
#135. Nested Layout với Route Group	167
#136. Bài tập design Login Form	168
#137. Chữa Bài Tập Design Login	169
#138. Modify login page with Social Media	169
#139. Modify login page with Credential	170
#140. Hiển Thị Thông Báo Lỗi Khi Login	170
Chapter 15: Module Tracks	171
#141. Bài tập hiển thị danh sách tracks (admin)	171
#142. Chữa bài tập hiển thị danh sách tracks (admin)	171
#143. Delete a track (admin)	171
#144. Ý Tưởng Design Upload Track	172
#145. Input Type File	173
#146. Tích hợp MUI Tabs	173
#147. Tích hợp Drag/Drop	174
#148. Bài tập Design Tab Information	176
#149. API Upload Single File	177
#150. Upload file with Axios	178
#151. Axios with Percents	178
#152. Upload track complete	179
#153. Add Custom Message	180
#154. Profile Page	181

#155. Bài tập render List images	181
#156. Customize track footer	182
#157. Quản lý React State Global	183
#158. Tích hợp React Context	184
#159. Play/Pause Track with Context	185
#160. Bài Tập Update view detail	187
Chapter 16: Module Likes, Comments	188
#161. Bài tập CRUD Comments (Admin)	188
#162. Bài tập hiển thị comment trên track	188
#163. Bài tập Component Comments	189
#164. Bài tập add a comment	190
#165. Bài tập add likes	191
#166. Add count view	192
Chapter 17: Module SEO (Search Engine Optimization)	193
#167. Điều cần biết về SEO	193
#168. Công cụ check SEO	194
#169. Metadata	195
#170. SEO 1: Tối ưu hóa project	196
#171. SEO 2: Đặt tên Title/Description (Static) cho web page	197
#172. SEO 3: Đặt tên Title/Description (Dynamic) cho web page	198
#173.1 Cách hosting ảnh FREE (Bonus)	199
#173.2 SEO 4: Share link website trên Facebook/Twitter	200
#174. SEO 5: Thêm logo cho website	202
#175. SEO 6: Url với slug	203
#176. SEO 7: Robot.txt, sitemap và manifest.json	204
#177. SEO 8: JSON-LD	205
#178. SEO 9: Next.js Image	206
Chapter 18: Advance - Tối ưu dự án Next.JS	210
#179. Add Progress Bar	210
#180. Handle Error	210
#181. Middleware	211
#182. Add Loading	212
#183. Nextjs Build Script	213
#184. Build với Docker	215
#185. Phân tích kết quả Next.js build	217
#186. SSR và Static Site Generation	218
#187. SSG và generateStaticParams	219
#188. NextJS Caching	220
#189.1 Revalidating data (Time based revalidation)	222
#189.2 Revalidating data (On-demand revalidation)	224
Chapter 19: Module Playlists	226
#190. Bài tập Playlist Page	226
#191. Bài tập Likes page	228
#192. Bài tập Chức năng search	229

#193. Login with Google	230
#194.1 Fix bug dự án	231
#194.2. Upgrade Next.js Version (và các thư viện khác)	232
Chapter 20: What's next ?	233
#195. Why NextJS ?	233
#196. Nhận xét về Level Fresher	234
#197. Nhận xét về khóa học này	235
#198. What's next - Học gì tiếp theo ?	236
Chapter 21: Update Next.js 14 với Antd (NEW)	237
#199. Chuyện công nghệ update	237
#200. Một vài lưu ý về Next.js Update	239
#201. Next.js 14 có gì hot ?	241
#202. Setup Next.js 14 Only	242
#203. Setup Antd	243
#204. Bài tập Design Login Form với Antd	243
#205. Khái niệm Flick giao diện	244
#206. Setup Antd với Render trên Server	245
Chapter 22: Server Actions (NEW)	246
#207. Tổng quan về Server Actions	246
#208. Handle Form với Server	247
#209. Loading State và useFormStatus Hook	248
#210. Handling Response Data và useFormState Hook	249
#211. Các công cụ sử dụng với Server Actions	250
#212. Call Server Actions from Functions ?	251
Chapter 23: Thực Hành Mutate Data với Server Actions (NEW)	252
#213. Design Giao Diện CRUD	252
#214. Bài Tập Create User Actions	253
#215. Bài Tập Update User Actions	253
#216. Bài Tập Delete User Actions	253
#217. Update Dự Án SoundCloud Sử Dụng Next.js 14	254
#218. So sánh Server Actions và Route Handler	255
#219. Tổng kết về Server Actions	257
Lời Kết	258

Chapter 0: Giới thiệu

Giới thiệu và demo kết quả đạt được sau khi kết thúc khóa học này.

#0. Demo Kết quả đạt được

Link video demo: <https://youtu.be/dsha7HgqMhl>

Về frontend, mình đã cover:

- Bootstrap (React Ultimate)
- Ant Design (React Test Fresher)
- MUI (React Pro Max & Next.js)

1. Demo thành quả đạt được

Dự án clone soundcloud (website nghe nhạc)

Tính năng chính:

- + Đăng nhập : hỗ trợ đăng nhập với Google/Github và tài khoản local của backend.
Có khả năng hỗ trợ đăng nhập với các nền tảng được liệt kê tại đây: <https://next-auth.js.org/providers/>
- + Nghe nhạc với:
 - audio track ở footer
 - wavetrack khi xem chi tiết
- + Comment theo thời gian của wavetrack
- + Tạo playlist/Like tracks
- + Với giao diện admin: CRUD user/tracks/comment...

2. Công nghệ sử dụng

Frontend:

- Client: Nextjs, MUI (**typescript**). Login with Next-auth (hỗ trợ đăng nhập Google/Github và Credential Provider (backend))
- Admin: Vite, Antd (typescript)

Backend: Nestjs (được cung cấp sẵn)

Về kiến thức React (typescript với Vite), ngoài kiến thức cơ bản, bổ sung thêm:

- React Children, React Fragment, React Context. Suspense/Lazy loading
- Các hook: useRef, useCallback, useMemo

Về kiến thức Nextjs:

- Nextjs 13 với App Router (typescript)
 - Nắm vững các kiến thức thay đổi của Nextjs:
 - + Routing (dynamic routes, route groups)
 - + Data Fetching (Server component vs Client component (use client))
 - + Authentication with Next-auth (sử dụng Session/JWT/Refresh Token)
 - + SEO (mức basic, test với lighthouse built-in extension)
 - + Rendering:
client side rendering (CSR), server side rendering (SSR),
static site generation (SSG), incremental Static Regeneration (ISR)
 - + Chế độ build:
 - build với docker
 - demo revalidate Tag after build :v
 - Sử dụng MUI để base component

3. Đối tượng theo học

- Bạn "bắt buộc" cần biết HTML, CSS và cú pháp javascript. Học viên bắt đầu từ số 0 theo học không phù hợp.

- Bạn không cần phải biết code Backend (vì backend được cung cấp sẵn).

Nếu được, bạn có thể xem qua 2 series: (hoàn toàn miễn phí trên Youtube Hỏi Dân IT)

- React cơ bản

- Typescript

-> 2 series trên sẽ giúp bạn có nền tảng vững chắc để theo học.

Phần còn lại, khóa học sẽ giúp bạn:

- Học kiến thức React.js (Typescript)

- Học Nextjs (Typescript)

Nếu bạn là người đã có tư duy lập trình (đã đi làm), cần 1 khóa học để end-game với React

=> đây chính là khóa học bạn cần.

Nếu chưa chắc chắn, cần tư vấn & hỗ trợ giải đáp thắc mắc, inbox Facebook Hỏi Dân IT:

<https://www.facebook.com/askITwithERIC>

4. Tài liệu khóa học

- Cung cấp qua pdf, được dùng để 'take note' kiến thức quan trọng, link source code, link ví dụ...

- Được đánh số version, link download trực tiếp trong khóa học

- Lưu ý về cách sử dụng bookmark khi xem online

5. Source code Frontend

Source code Frontend cả khóa học "không được" cung cấp. Bạn cần xem video và code theo. Cuối mỗi video đều có phần review các files thay đổi.

Chỉ có những video nào, mình nói cho source code (thông thường là các video khó), mình sẽ để link download source code trong tài liệu pdf.

Lý do: 99% những bạn xin full source code, thông thường sẽ không xem video => review và đánh giá khóa học kém chất lượng.

100% code theo videos khóa học sẽ thành công, vì nếu có lỗi, chỉ cần bạn report, mình sẽ làm video update.

6. Source code Backend

- **Source code backend bạn sẽ được cung cấp sẵn, dưới dạng mã hóa.** Chỉ chạy và không sửa đổi.

Lý do: Đây là khóa học frontend, nên sẽ không học backend.

Bạn muốn biết source code này viết như nào, vui lòng học lộ trình backend

7. Chuyện deploy

- Khóa học này chạy localhost, vì hosting free ngoài kia rất "lởm". hàng free mà.
- Do khóa học này có backend và việc thao tác với file => hosting free không có khả năng lưu trữ file
- Deploy chuyên nghiệp cần biết cả frontend và backend
- Trong cv, bạn ghi link github là đủ. nếu cần thiết, thì quay video demo, thay vì deploy free
- Muốn deploy thực tế, bạn cần học kỹ năng backend, sau đây deploy với server trả phí.
Link khóa học deploy: <https://hoidanit.com.vn/khoa-hoc/ultimate-guide-to-deploy-react-nodejs-640bee82f7099c369b3bc6a4.html>

8. Chuyện bản quyền/học chung

Khóa học được đánh bản quyền theo thỏa thuận giữa bạn và tác giả Hỏi Dân IT.

Mọi hành vi "share/mua chung" đều được coi là vi phạm => đồng nghĩa với việc bạn đang tước bỏ đi quyền lợi của học viên

Nếu bạn có nhu cầu học chung (mua chung), cứ inbox mình. Chúng ta sẽ tìm được tiêng nói chung (mình có hỗ trợ khi mua với số lượng lớn)

9. Hình thức mua

- Inbox trực tiếp fanpage Hỏi Dân IT: <https://www.facebook.com/askITwithERIC>

- Luôn có chính sách ưu đãi cho học sinh/sinh viên/người chưa đi làm và các học viên cũ

#1. Hướng Dẫn Download Tài liệu khóa học

Tài liệu khóa học sẽ được update theo thời gian và được "đánh version".

Nếu trong quá trình khóa học, bạn thấy tài liệu không đầy đủ, thì check lại video này nhé :v

#2. Yêu cầu của khóa học

1. Biết HTML, CSS và cú pháp Javascript (tự học)

Về HTML, CSS => tự học

Về Javascript, tham khảo: (chỉ cần học để hiểu cú pháp khai báo của javascript)

<https://www.w3schools.com/js/>

Link khóa học Javascript miễn phí:

<https://www.youtube.com/playlist?list=PLncHg6Kn2JT5dfQqpVtfNYvv3EBVHHVKo>

2. Biết cú pháp typescript:

Link khóa học Typescript miễn phí:

<https://www.youtube.com/playlist?list=PLncHg6Kn2JT5emvXmG6kgeGkrQjRqxsb4>

3. Biết sử dụng Git để quản lý mã nguồn

Link khóa học Git miễn phí:

<https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo>

#3. Về khóa học này

Mục tiêu:

- Chỉ 1 khóa học duy nhất, học React & Next.JS

- Giúp định hướng và rút ngắn thời gian học kiến thức, phục vụ mục đích đi thực tập & đi làm

Khóa học này có "bị trùng nội dung" với lộ trình FE đã có của mình không ?

Câu trả lời là không.

Điểm khác biệt của khóa học này, là học thêm Next.js

Tuy nhiên, để học Nextjs, bạn cần "biết React trước".

=> các khóa cũ, là học "mình React". điều này không có trong khóa Next.js (từ kiến thức cơ bản tới nâng cao, cho tới việc dùng với Typescript)

=> học xong khóa next.js này, recommend (gợi ý) học thêm các khóa FE còn lại để hiểu sâu hơn về React :v

4. Về chuyện leak khóa học và mua lậu

Mình biết rất nhiều bạn khi học khóa học này của mình, là mua lậu qua bên thứ 3. chuyện này là hoàn toàn bình thường, vì thương hiệu "Hồi Dân IT" đang ngày càng khẳng định được vị thế của mình.

Nhiều bạn hỏi mình, sao mình không 'chặn việc mua lậu'. nói thật, nếu mình làm, là làm được đấy, cơ mà nó sẽ gây ra sự bất tiện cho học viên chân chính (con sâu làm rầu nòi canh). Với lại, ngay cả hệ điều hành windows, còn bị crack nữa là @@

Mình cũng có 1 bài post facebook về chuyện này:

<https://www.facebook.com/askitwitheric/posts/pfbid02gyasktd3semgxat6nevnvwh4c8epzu3i7kpzhr7s7gmmfcvucyz96eb8avnvgwnhl>

Với các bạn học viên chân chính, mình tin rằng, những cái các bạn nhận được từ mình khi đã chấp nhận đầu tư, nó sẽ hoàn toàn xứng đáng. vì đơn giản, với cá nhân mình, khách hàng là thượng đế.

VỚI CÁC BẠN MUA LẬU, MÌNH CHỈ MUỐN CHIA SẺ THẾ NÀY:

1. TRÊN ĐỜI NÀY, CHẮNG CÓ GÌ CHẤT LƯỢNG MÀ MIỄN PHÍ CẢ.

VIỆC BẠN MUA LẬU QUA BÊN THỨ 3, LÀ GIÚP BỌN CHÚNG LÀM GIÀU VÀ GÂY THIỆT HẠI CHO TÁC GIẢ.

NẾU NHÌN VỀ TƯỞNG LAI => CÀNG NGÀY CÀNG ÍT TÁC GIẢ LÀM KHÓA HỌC => NGƯỜI BỊ HẠI CUỐI CÙNG VẪN LÀ HỌC VIÊN

2. HÃY HỌC THÓI QUEN TRÂN TRỌNG GIÁ TRỊ LAO ĐỘNG

NÓ LÀ THÓI QUEN, CŨNG NHƯ SẼ LÀ MỘT PHẦN TÍNH CÁCH CỦA BẠN.

ĐỪNG VÌ NGHÈO QUÁ MÀ LÀM MẤT ĐI TÍNH CÁCH CỦA BẢN THÂN.

NẾU KHÓ KHĂN, CỨ INBOX MÌNH, MÌNH HỖ TRỢ. VIỆC GÌ PHẢI LÀM VẬY =))

3. MÌNH ĐÃ TỪNG LÀ SINH VIÊN GIỐNG BẠN, MÌNH HIỂU TẠI SAO CÁC BẠN LÀM VẬY. HÃY BIẾT CHO ĐI. SỐNG ÍCH KỶ, THÌ THEO LUẬT NHÂN QUẢ ĐẤY, CHẮNG CÓ GÌ LÀ NGẦU NHIÊN CẢ

4. NẾU BẠN THẤY KHÓA HỌC HAY, HÃY BIẾT DONATE ĐỂ ỦNG HỘ TÁC GIẢ. LINK DONATE: <https://hoidanit.com.vn/donate>

Hành động nhỏ nhưng mang ý nghĩa lớn. Hãy vì 1 cộng đồng IT Việt Nam phát triển. Nếu làm như các bạn, có lẽ chúng ta đã không có Iphone, không có Apple như ngày nay rồi @@

#4.1 Hướng Dẫn Sử Dụng Khóa Học Hiệu Quả

Source code frontend phần React Vite (hết video #76):

https://drive.google.com/file/d/1RAWwfB1bqQR_mdmExChIihPByiETpg6j/view?usp=sharing

Lưu ý: với source code trên, các bạn cần tự setup và xem giúp mình chapter 5 để biết cách chạy backend cho dự án

Hoặc: có thể sử dụng source code trên để đỡ phải code, trong khi vẫn xem đầy đủ video của khóa học

Mục đích cung cấp source code, là dành cho các bạn “**đã biết code React với Typescript**” (ví dụ như là đã học qua lộ trình frontend của mình rồi chẳng hạn).

Tất cả trường hợp khác, mình KHÔNG KHUYẾN KHÍCH sử dụng, thay vào đấy là học theo trình tự của khóa học

=> kiến thức được liên mạch và tránh bị hổng kiến thức

#4.2 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

Cài đặt các công cụ cần thiết cho khóa học

#5. Công cụ code IDE

Cài đặt Visual Studio Code (VSCode):

Link download: <https://code.visualstudio.com/download>

- Setup VSCode: auto format on save

- Setup extension: (mục đích setup extension là "hỗ trợ" việc code, không phải là "phụ thuộc" vào extension)

=> chỉ setup extension "cần thiết", và hạn chế tối đa sự phụ thuộc vào extension

+ auto complete tag, auto close tag, auto rename tag

=> hỗ trợ việc code html nhanh hơn

+ code spell checker

=> hỗ trợ check tên biến/hàm có bị sai chính tả hay không ? (lưu ý là check tiếng anh, ko phải là check tiếng việt)

- Lưu ý về extension eslint, pretty (nếu cài: disabled)

Nếu bạn "từng học ở các nguồn khác", họ hướng dẫn cài đặt eslint, rồi pretty để "code cho đẹp hơn", thì vui lòng disabled những extension này, để tránh những "báo lỗi" không cần thiết.

Eslint, hay pretty, là cách bạn đặt ra quy tắc code, nhưng mà, mỗi cá nhân, mỗi công ty, có 1 quy định khác nhau

=> nếu bạn muốn setup eslint, hay pretty, thì setup trực tiếp trong project, không setup qua extension bạn nhé

Ở đây, chúng ta disabled, mục đích là dùng "những format" được quy định sẵn khi dùng VScode (không cần cài đặt extension).

#6. Browser

Về browser (trình duyệt web), bạn vui lòng tuân theo quy định bên dưới nhé :

1. Sử dụng Google Chrome (bắt buộc):

Download: <https://www.google.com/chrome/>

Lưu ý không dùng Cốc Cốc, Firefox hay Edge..., vì:

+ trong khóa học, mình sử dụng Google Chrome. Dùng giống nhau, sẽ có giao diện code giống nhau => hạn chế lỗi "không cần thiết"

+ **Google Chrome phổ biến nhất**, đi làm, chúng ta test cũng ưu tiên Google Chrome đầu tiên (cứ phổ biến mà xài)

Fact: Nếu bạn để ý, bạn sẽ thấy Firefox, hay Edge, đều bắt chước tính năng dành cho developer của Google Chrome :v

2. Chuyển ngôn ngữ Tiếng Anh (bắt buộc):

Bạn nào dùng ngôn ngữ tiếng việt => vui lòng chuyển qua tiếng anh, vì:

+ Giao diện ngôn ngữ là khác nhau. Các "thuật ngữ" ngành IT, dịch sang tiếng việt, nó không sát nghĩa (không đúng)

+ Đi làm thực tế, không ai dùng tiếng việt đâu bạn. sự thật đấy (nếu bạn không muốn được gọi là đứa nhà quê, hai lúa :v)

+ dùng tiếng anh, nó có lợi ích về lâu dài (giúp bạn đi nhanh hơn, vì tài liệu = tiếng anh luôn có sẵn)

+ Nếu bạn chưa biết gì về tiếng anh => dùng google translate.

Ở đây, chưa biết gì, thì code theo

code nhiều thành quen thỏi. giống bạn chơi game ấy, toàn từ ngữ tiếng anh, sao bạn chơi được ? (học code nó tương tự bạn nhé)

3. Hướng dẫn chuyển đổi ngôn ngữ của Google Chrome

//todo (trong video đã hướng dẫn)

#7. Node.JS

1. Node.JS là gì ?

Bạn học React, tại sao lại cần Node.JS ?

Node.js là **platform** giúp bạn có thể thực thi ngôn ngữ Javascript trên máy tính (server)

Điều này tương tự với:

Bạn muốn học **Microsoft Words** => bạn cần cài đặt hệ điều hành **Windows**
(Bạn học **React** => cần cài đặt **Node.JS**)

như vậy, platform là "môi trường" giúp bạn chạy code.
chúng ta không học "node.js", mà học "react".

Node.js là điều kiện "bắt buộc" để chúng ta chạy code các bạn nhé.

2. Cài đặt Node.JS

Lưu ý: không cài đặt version Node.js mới nhất, vì rất nhiều thư viện chưa kịp hỗ trợ Node.js latest

Việc cài chính xác 1 version của Node.js, sẽ giúp code của mình và bạn giống nhau 100% (hạn chế tối đa lỗi không cần thiết)

Điều này tương tự với việc, bạn "choi 1 game trên windows 7 rất ok", nhưng điều này "không chắc chắn" khi bạn dùng windows 11, right ?

- Cài node v18.17.0: <https://nodejs.org/download/release/v18.17.0/>

- Muốn dùng nhiều version (tránh ảnh hưởng tới dự án cũ => nvm):

https://www.youtube.com/watch?v=ccjKHLyo4IM&list=PLncHg6Kn2JT6E38Z3kit9Hnif1xC_9Vql&index=40

3. Kiểm tra cài đặt Node.js

dùng câu lệnh: **node -v**

khi cài đặt node.js => đồng nghĩa với việc bạn cài đặt npm (node package manager) =>
công cụ giúp cài đặt thư viện code js
=> kiểm tra bằng cách: **npm -v**

#8. Quản lý code với Git

Nếu bạn chưa biết gì về Git, vui lòng xem nhanh tại đây:

<https://www.youtube.com/playlist?list=PLncHg6Kn2JT6nWS9MRjSnt6Z-9Rj0pAlo>

Mục đích sử dụng Git:

- "Kéo code" dự án thực hành
- Lưu trữ "backup" cho code. Tránh tình huống máy tính bị hư => mất hết code
- Git là "công cụ bắt buộc phải biết" khi đi làm thực tế (đi thực tập/fresher)

Chapter 2: Ôn tập Javascript

Ôn tập các kiến thức nền tảng của javascript hay được sử dụng với React. Mục tiêu của chương, là khi nào bạn cần, sẽ giải đáp ngay lập tức thắc mắc của bạn.

#9. Cách học kiến thức Javascript hiệu quả

Mục tiêu của chương, là giúp bạn "nhìn mặt chữ & nhận dạng", không phải là giúp bạn thực hành ngay lập tức

Chương này không phải học lại javascript từ đầu, mà là định hướng các cú pháp "hay dùng" (thậm chí là đi làm) bạn hay gặp phải.

- Sau này học react, nếu bạn quên, có thể xem lại để hiểu hơn. Vì suy cho cùng, React cũng là javascript mà thôi
- Không ai có thể hiểu & vận dụng tất cả mọi thứ ngay từ đầu.

Tất cả đều cần "thực hành"

Nếu bạn quên, thì bỏ qua. Miễn sao nhớ được tên "của kiến thức" là được.

sau đây xem lại => thực hành

Chỉ có thực hành "đủ nhiều", mới thấy nó dễ.

#10. Destructuring Assignment

Tài liệu: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment

Destructuring Assignment là cách chúng ta giản lược cú pháp viết code (giúp viết code ngắn & thuận tiện hơn) khi thao tác với **array** và **object**

1. Object Destructuring

Tài liệu: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment#object_destructuring

Ví dụ:

```
const user = {  
    name: "Eric",  
    age: 25,  
    address: "Ha Noi"  
}
```

before:

với cách viết truyền thống, để lấy ra 3 biến name, age và address, cần sử dụng:

```
const name = user.name;  
const age = user.age;  
const address = user.address;
```

=> càng nhiều biến (variable), số lượng dòng code sẽ càng nhiều.

after:

```
const { name, age, address } = user;  
=> sử dụng duy nhất đúng 1 dòng. Chúng ta dùng const {} vì user là một object.  
keyword const giúp khởi tạo kiểu giá trị (type) cho các biến bên trong {}
```

Với object destructuring => sử dụng dấu {}

2. Array Destructuring

Tài liệu: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Destructuring_assignment#array_destructuring

Ví dụ:

```
const level = ["internship", "fresher", "junior", "middle", "senior"]
```

before:

```
const internship = level[0]
const senior = level[4]
```

after:

```
const [internship, fresher, junior, middle, senior] = level;
```

=> tương tự như object destructuring, điểm khác biệt duy nhất là dùng [] thay vì { }

//Part 2

3. Ví dụ với React

Lưu ý: không cần hiểu props, useState là gì (vì sau này sẽ hiểu), điều quan trọng là bạn nhận ra ý nghĩa của cách viết code.

Ví dụ 1:

```
function User (props) {
    const {name, age, address} = props;
    // nhìn vào cách viết này, bạn sẽ hiểu props là 1 biến object, và chúng ta đang sử dụng
    // object destructuring
```

//thay vì viết:

```
const name = props.name;
const age = props.age;
....
```

```
}
```

Ví dụ 2:

```
function Level (){
```

```
    const [level, setLevel] = useState(...)
```

//đây là array destructuring. Không cần biết useState() là cái gì, cơ mà chúng ta biết nó là 1 array

và ở đây:

level === useState()[0] => phần tử đầu tiên của array

setLevel === useState() [1] => phần tử thứ 2 của array

```
}
```

#11. Spread syntax

Tài liệu:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Spread_syntax

Hiểu 1 cách đơn giản, spread syntax (toán tử mở rộng) giúp bạn có thể "copy" đối tượng (array, object).

1. Sử dụng với array

```
const arr1 = [1, 2, 3]
```

muốn thêm phần tử 4 (vị trí thứ 4), và gán nó vào ar2 thì làm sao ?

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/push

before:

```
const arr1 = [1, 2, 3];
arr1.push(4)
```

```
const arr2 = arr1;
```

lưu ý: không viết là const arr2 = arr1.push(4)

(vì hàm push trả về length của arr => arr2 = length new array)

after:

```
const arr2 = [ ...arr1, 4 ] => thêm phần tử vào cuối mảng
```

```
const arr2 = [ 4, ...arr1 ] => thêm phần tử vào đầu mảng (arr.unshift)
```

2. Sử dụng với object

```
const obj1 = { foo: "bar", x: 42 };
const obj2 = { foo: "baz", y: 13 };
```

```
const clonedObj = { ...obj1 };
// { foo: "bar", x: 42 }
```

```
const mergedObj = { ...obj1, ...obj2 };
```

```
// { foo: "baz", x: 42, y: 13 }
```

Lưu ý: không nên làm dụng spread syntax, vì đây là shallow copy, không phải là deep copy

Tham khảo: <https://hoidanit.com.vn/khoa-hoc/react-nang-cao-lieu-ban-da-hieu-react.html?id=640be92bf7099c369b3bc685>

Ví dụ 1: (ok)

```
const first_person = {  
  name: "Jack",  
  age: 24,  
}  
  
const second_person = { ...first_person };  
  
second_person.age = 25;  
  
console.log(first_person.age); // output: 24  
console.log(second_person.age); // output: 25
```

Ví dụ 2: (not ok - sử dụng nested object)

```
const first_person = {  
  name: "Jack",  
  age: 24,  
  address: {  
    apartment: "A",  
    city: "London"  
  }  
};  
  
const second_person = { ...first_person };  
  
second_person.age = 25;  
second_person.address.apartment = "N";  
console.log(first_person.address.apartment); // output: N  
console.log(second_person.address.apartment); // output: N
```

#12. Conditional Operator (Toán tử điều kiện)

Tài liệu: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Conditional_operator

Toán tử điều kiện tương tự như câu điều kiện if, gồm 3 phần:

condition ? exprIfTrue : exprIfFalse

Nếu "điều kiện" là đúng, <exprIfTrue> sẽ được thực thi. ngược lại, <exprIfFalse> sẽ được thực thi.

Việc sử dụng toán tử điều kiện, again, giúp code của bạn ngắn hơn thôi :v

Ví dụ:

before:

```
const a = 10;
let b = null;
if(a > 5){
  b = true;
} else {
  b= false;
}
```

after:

```
a = 10;
const b = a > 5 ? true : false;
```

```
const age = 26;
const beverage = age >= 21 ? "Beer" : "Juice";
console.log(beverage); // "Beer"
```

#13. Optional Chaining

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Optional_chaining

Optional Chaining là các truy cập thuộc tính của object (hoặc function) mà không muốn ném ra lỗi (nếu có)

Lưu ý: Không nên lạm dụng cách làm này (chỉ dùng, khi bạn biết bạn đang làm gì)

1. Sử dụng với variable

```
const adventurer = {  
  name: 'Alice',  
  cat: {  
    name: 'Dinah'  
  }  
};
```

before:

```
const dogName = adventurer.dog.name;  
console.log(dogName); //error  
=> throw an error
```

after

```
const dogName = adventurer?.dog?.name;  
console.log(dogName); //undefined
```

2. Sử dụng với function

Ví dụ:

```
const a = undefined;
```

```
a.forEach(item => ...)
```

```
a?.forEach(...)
```

Part 2:

3. Gán giá trị mặc định

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Operators/Nullish_coalescing

Chapter 3: React Basic

Học các kiến thức cơ bản và cốt lõi nhất của React thông qua các ví dụ thực tế.

#14. Lưu ý về cách code React

1. Phong cách code (free styles)

- Cách đặt tên biến
- Cách khai báo function
- Cách dùng dấu chấm, dấu phẩy (dùng dấu chấm phẩy cuối mỗi dòng code, tương tự như code java => tránh lỗi không cần thiết)
- không dùng extension (hạn chế tối đa nhất có thể)

=> mình không phải là "master code", tuy nhiên, mình đã trải qua cơ sở công ty, và mỗi công ty đều có "**convention**" khi code.

Vì vậy, nếu bạn là beginners, tốt nhất hãy tập code theo mình. Đิ làm, người ta chỉ yêu cầu hơn, chứ không kém bạn nhé.

2. Tài liệu học React

- Về trang tài liệu chính thống:

Từ tháng 4/2023: <https://react.dev/>

Trước tháng 4/2023 : **reactjs.org or legacy.reactjs.org**

=> tài liệu trên website này không còn được update nữa.
tuy nhiên, các **core function (kiến thức React)** luôn không đổi.

Phần update (khác biệt giữa website "mới" và "cũ"), là cách trình bày tài liệu.

cũng như các tính năng mới của react (ví dụ như react action, react server...) sẽ không có trên trang tài liệu cũ.

#15. React là gì ? Phân loại dự án với React

Part 1: React là gì

React là “library” (thư viện) viết bằng ngôn ngữ Javascript, giúp việc code UI (giao diện web) trở nên dễ dàng và tiện lợi hơn

Library: tập hợp các khối code (block code/ functions)

Framework: thiên về cách tổ chức code (architecture/kết cấu), đồng thời đã làm sẵn rất nhiều tính năng.

Platform: Môi trường chạy code

React là “library” javascript => kiến thức về javascript bạn cần biết trước. Bạn code React, “bản chất” là bạn đang code ngôn ngữ javascript

React được Facebook chống lưng và phát triển. Amazing

1. So sánh React, Angular và Vue

<https://npmtrends.com/@angular/core-vs-react-vs-vue>

Lựa chọn React, vì nó phổ biến nhất

=> nguồn tài liệu phong phú, cộng đồng support nhiều

2. Chuyện 1m2 20 ông React :v

Câu chuyện này tương tự, doanh nghiệp luôn than thở, “lúc nào cũng cần/thiếu nhân sự”, trong khi người tìm việc không kiếm được việc làm.

Ở đây, bạn nên hiểu, “doanh nghiệp cần”, là cần người “làm được việc”

1m2 20 ông React, cơ mà ông nào cũng giống ông nào, chỉ biết basic (ví dụ viết Hello world), thất nghiệp là chuyện “quá bình thường”

Part 2:

1. Lịch sử ra đời

Hiện nay, react có 2 loại chính là: **React ở client** và **React ở Server**

+ React ở Client: create-react-app, vite

+ React ở Server: next.js, remix, gatsby

Why ?

+ **Trước 2012**, khi "các library/framework" chuyên làm FE chưa ra đời (React, Angular, Vue), website thông thường là full-stack.

điều này có nghĩa là, bạn code nguyên 1 website, sử dụng đúng 1 ngôn ngữ (java, php, C#...)

Nhược điểm của cách làm này, là code FE "rất khổ", vì đa phần là code thuần html với "view engine"

=> tính mở rộng và tái sử dụng code không cao (UI), trải nghiệm của người dùng "không sướng" (UX)

ví dụ về UX: chat real-time

Với React: (ra đời chính thức vào năm 2013)

Mục tiêu của React là giúp code FE "sướng hơn và nhàn hơn" cách code HTML truyền thống.

+ **Từ 2012 tới 2022**: sự phát triển của React (phía client)

+ **Từ 2022 -> nay (trending)**: React phía server

=> giống hệt như thời trước 2012.

Điểm khác biệt, là "bạn không cần reload trang web để xem sự thay đổi :v)

2. React ở Client

- Không đặt nặng về SEO
- Quan trọng việc "thay đổi dữ liệu thường xuyên" ở client
- Dữ liệu sẽ được 100% client xử lý => loading khi vào trang

ví dụ: <https://iboard.ssi.com.vn/>

3. React ở Server

- Quan tâm tới SEO
- Dữ liệu ít thay đổi
- Dữ liệu sẽ được server xử lý => không bị loading khi vào trang

ví dụ: <https://hoidanit.com.vn>

4. Chia sẻ câu chuyện website Hỏi Dân IT

- code html thuần (2h)
- code với react vite (client) (2 tuần)
- code với next.js (server) (1 tháng) => thời gian lâu hơn, vì có sự thay đổi về thư viện (antd => mui)

#16. Setup Project React với Vite

Chúng ta sẽ học cả React ở client và React ở server
=> đi công ty nào dùng React cũng sống được

Bắt đầu từ React client, rồi tới React server (chính là lịch sử phát triển của React)
=> hiểu tại sao công nghệ nó lại thay đổi.

1. Setup dự án

Tài liệu: <https://vitejs.dev/guide/>

Lưu ý: chỉ làm theo link tài liệu, khi bạn "đã có kiến thức về react", hiểu rõ bạn đang làm gì.

Link source code: <https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/react-vite-starter>

Lưu ý: đã cấu hình eslint để check lỗi (react overlay):

<https://github.com/vitejs/vite/issues/2076>

2. Run Hello World

+ cài đặt thư viện cần thiết: npm i (hoặc gõ đầy đủ: npm install)

Lưu ý về message thông báo:

<https://stackoverflow.com/questions/69885467/i-have-npm-vulnerability-issue>

added 149 packages, and audited 150 packages in 33s

37 packages are looking for funding
run `npm fund` for details

found 0 vulnerabilities

+ chạy dự án với câu lệnh: npm run dev

Mặc định, project vite chạy tại địa chỉ: http://localhost:5173

#17. Cấu trúc dự án React Vite

1. Folder:

- node_modules: đây là nơi "code của thư viện cài đặt". Tất cả thư viện được sử dụng trong dự án sẽ được lưu tại đây
- public: như tên gọi (công khai) => đây là nơi bạn lưu trữ file và public ra ngoài (ví dụ css/js/images...)
- src : đây là nơi lưu code dự án (không public ra ngoài)

2. Files

vite.config.ts => file cấu hình dự án với Vite
tsconfig.json và tsconfig.node.json => cấu hình typescript cho dự án
README.md => guide

package.json => đây là file cấu hình tất cả câu lệnh chạy dự án, cũng như thư viện cài đặt

Nguyên tắc: cài thêm 1 thư viện mới => update file package.json

package-lock.json: file lưu chi tiết thông tin thư viện cài đặt (version cài đặt)

.gitignore : quy định các file không đẩy lên git

.eslintrc.cjs: cấu hình eslint (phục vụ mục đích check code)

index.html : đây là file project sẽ chạy

<div id="root"></div>

=> nội dung code React sẽ được dịch và chèn vào div root ở trên

#18. Think in React

Tài liệu:

<https://react.dev/learn/thinking-in-react>

- Khái niệm component, why ?
- Chia nhau layout, giống chơi lego
- Giúp tái sử dụng code
- Component >< HTML ở chỗ, có thể code được javascript bên trong html

#19. Component

Tài liệu: <https://react.dev/learn/your-first-component>

Lưu ý: viết hết trong file main app (chưa sử dụng cú pháp import/export)

1. Định nghĩa component

- Component, suy cho cùng, là javascript function

Tên file: .js , .jsx (dùng với javascript)

.ts, .tsx (dùng với typescript)

Gồm 2 thành phần:

+ logic javascript

+ template (jsx)

=> Component >< HTML ở chỗ, có thể code được javascript bên trong html

2. Nguyên tắc Khi báo function :

- Tên function cần viết hoa chữ cái đầu tiên

- cần có hàm return (template/html) để biến nó thành react component

```
MyFunction(){  
  return(  
    <div> </div>  
  )  
}
```

Part 2:

Lưu ý về cách viết code:

1. không dùng cách viết inline

```
return ;
```

2. luôn sử dụng dấu () với hàm return

nên nhớ, component là function => return () có nghĩa là function này "chắc chắn/always) trả về giá trị bên trong ()

before:

```
return ;
```

after:

```
return ();
```

#20. Import/Export Component

Tài liệu:

<https://react.dev/learn/importing-and-exporting-components>

Về cú pháp export:

<https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/export>

Import/Export là cách "reuse - tái sử dụng" functions/variables giữa các "file" của javascript

Ví dụ:

```
// 📁 say.js
function sayHi(user) {
  alert(`Hello, ${user}!`);
}

function sayBye(user) {
  alert(`Bye, ${user}!`);
}

export {sayHi, sayBye}; // a list of exported variables
```

```
// 📁 main.js
import {sayHi, sayBye} from './say.js';

sayHi('John'); // Hello, John!
sayBye('John'); // Bye, John!
```

=> lưu ý về **export default**

React Component cũng là function, nên cũng tuân theo cú pháp import/export của javascript

1. Cú pháp Import/Export

- Tách riêng component là 1 file

- export function

```
const MyFunction(){
    return (
        //todo
    )
}

export default MyFunction;
```

--

```
import MyFunction from ...
```

2. Về cách viết code

Khi sử dụng export default và export nhiều variables/functions:

Default : `export default function Button() {}` `import Button from './Button.js';`

Named `export function Button() {}` `import { Button } from './Button.js';`

#21. React JSX

Tài liệu: <https://react.dev/learn/writing-markup-with-jsx>

- Component >< HTML ở chỗ, có thể code được javascript bên trong html

JSX là cũ pháp giúp viết code html bên trong file javascript (.js) => .jsx

1. Convert HTML sang JSX

```
<h1>Hedy Lamarr's Todos</h1>

<ul>
  <li>Invent new traffic lights </li>
  <li>Rehearse a movie scene </li>

  <li>Improve the spectrum technology
</ul>

export default function TodoList() {
  return (
    // ???
  )
}
```

nó không hoạt động, vì không tuân theo đúng nguyên tắc của JSX

2. Nguyên tắc của JSX

- Luôn trả ra 1 single root element
- Đóng tất cả các tags . ví dụ <input/>
- Sử dụng className, thay vì class (vì keyword class bị trùng trong file javascript)

#22. Sử dụng Variables với JSX

Tài liệu:

<https://react.dev/learn/javascript-in-jsx-with-curly-braces>

Javascript có các kiểu dữ liệu chính:

Nhóm 1:

string

number

Nhóm 2:

object

array

Boolean => render nothing

null/undefined => render nothing

Để sử dụng biến trong JSX, sử dụng { variable_name }

Lưu ý ngoại lệ là thuộc tính (attribute) style => sử dụng CSS inline

#23. Bài tập design Component Input

Design giao diện theo yêu cầu:

- Xóa tất cả giao diện đang có, xóa import file css (để không vỡ giao diện)
- Tạo component **InputTodo**
- Viết cú pháp jsx để tạo giao diện

Các bước thực hiện:

- Tạo component mới (InputTodo)
- Import vào file **App.tsx**

Lưu ý:

Về cách đặt tên file extension component

.jsx => code javascript với React

.tsx => code typescript với React

.js => file javascript thông thường

.ts => file typescript thông thường

Tên file đặt tùy thích (viết hoa, viết thường thoải mái), **nhưng tên component phải viết hoa chữ cái đầu tiên.**

Không nhất thiết phải **import React from 'react'** => cách nhiều khóa học cũ dạy

Với react 18, khi viết code javascript trong file .jsx or tsx => không cần import React

#24. Props là gì

Tài liệu: <https://react.dev/learn/passing-props-to-a-component>

1. Phân loại component

- Cha (parent)

- Con (child)

2. Props

Props, là viết tắt của property (properties)

translate: thuộc tính

feeling: tài sản kế thừa từ cha => con

Props là cách thức các component giao tiếp với nhau, giúp component cha truyền dữ liệu xuống component con

Gồm 2 bước:

1. Pass props to child component (truyền props xuống component con)

```
export default function Profile() {  
  return (  
    <Avatar  
      person={{ name: 'Lin Lanying', imageUrl: '1bX5QH6' }}  
      size={100}  
    />  
  );  
}
```

2. Read props (from child component)

```
MyFunction (props) {  
  ...  
}
```

#25. Props với Typescript

1. Tại sao cần typescript

- Được check và gợi ý code (chấm hết :v)

2. Sử dụng với Typescript

- Định nghĩa type với interface https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/basic_type_example/#types-or-interfaces

Tham khảo : <https://hoidanit.com.vn/khoa-hoc/react-pro-typescript-thuc-hanh-du-an-portfolio.html?id=6458f8f188a9e90d1174e47d>

- Code typescript thực chất là javascript + định nghĩa type

3. Lưu ý khi viết code cho beginners

Do: MyFunction (props)

Don't: MyFunction({ })

#26. React Devtool

Tài liệu: <https://react.dev/learn/react-developer-tools>

Download: <https://chrome.google.com/webstore/detail/react-developer-tools/fmkadmapgofadoplbjfkapdkoienihi>

#27. Khái niệm Re-render

Thay đổi props (sử dụng devTool)

=> component re-render

#28. Rendering Lists

Để sử dụng vòng lặp với React bên trong JSX, chúng ta **sử dụng map** (thay vì for, while)

Về vòng lặp map: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Array/map

Sử dụng map, thay vì vòng lặp for, hay while (do while) trong JSX vì:

- Vòng lặp map “trả ra” một mảng mới, không làm ảnh hưởng tới giá trị cũ
- Sử dụng for, hay while, chỉ đơn thuần là “kiểm tra các phần tử”, không tạo ra mảng mới (data mới), và có thể làm ảnh hưởng tới giá trị cũ

=> **Nguyên tắc đơn giản với JSX của React, là dùng map** để lặp các phần tử (thay vì for, while)

#29. Key

Lưu ý: warning khác với error, mặc dù đều có chữ màu đỏ (chữ màu đỏ thường không tốt, right ?)

Warning, đa phần là không ảnh hưởng tới hệ thống, bạn fix được thì hiệu năng ứng dụng cao, không fix được cũng chẳng sao

Error: đa phần là lỗi cần fix (vẫn có ngoại lệ, là có lỗi xảy ra, hệ thống vẫn chạy bình thường, về ví dụ minh họa, các video tiếp theo sẽ có bạn nhé).

1. React Key

React key, là các React nhận biết các phần tử HTML trên màn hình.

React dùng key (tương tự chúng ta dùng căn cước công dân), để định danh từng phần tử HTML.

Mục đích định danh này, là mỗi khi cập nhật (quá trình re-render), **react chỉ cần cập nhật “chính xác phần tử đấy”, mà không cần phải “vẽ lại/re-render” cả trang web**, như vậy hiệu năng ứng dụng sẽ cao.

2. Tại sao không dùng Id, mà lại dùng key

Khi chúng ta sử dụng React, tư duy cần phải theo hướng “component”

Với id, trên 1 website, không thể có 2 phần tử với cùng ID, nếu có, sẽ gây ra lỗi.

Với key và tư duy component, trên 1 website có thể tồn tại bao nhiêu component cũng được (không giới hạn số lượng).

Và nếu trùng key cũng không sao (vì React rất thông minh, nó sẽ biết key nào, ứng với component nào, chứ không đơn thuần dựa vào tên của key)

Chapter 4: React Hook

Học kiến thức react "hiện đại" bằng cách sử dụng Hook

#30. Event Handler

<https://react.dev/learn/responding-to-events>

Tất cả các event thông thường của HTML, React đều hỗ trợ.

Các event hay dùng nhất với React, có thể kể tới như click, change...

Lưu ý về cách code:

Bên trong hàm event, sử dụng Arrow function

Ví dụ:

Không code như này:

onClick={ handleClick }

Mà Code như này:

onClick={ () => handleClick() }

Lý do: sử dụng arrow function giúp tránh các lỗi sai không cần thiết, đồng thời giúp hiệu năng cao hơn.

(chi tiết mình giải thích trong phần nâng cao của khóa React Ultimate)

<https://hoidanit.com.vn/khoa-hoc/react-ultimate-react-co-ban-tu-z-toi-a.html?id=640b50b96cc592d780aab976>

#31. State là gì ?

<https://react.dev/learn/state-a-components-memory>

1. Khi cách dùng thông thường là chưa đủ

- Thay đổi biến => giá trị được thay đổi => nhưng giao diện không đổi (không re-render)
- Ví dụ: tạo global vars => update

2. Ví dụ về state

- State (trạng thái), dùng để hiển thị "trạng thái" của component
- State là 1 biến đặc biệt (chỉ React hiểu nó), không share giữa các component (nghĩa là, phạm vi của nó là local)
- State thay đổi => giao diện thay đổi (re-render)

#32. useState Hook

1. Why hook

Hook là cách React có thể tương tác với component (function)

Hook hỗ trợ từ **version React 16.8** => React version cũ, sẽ dùng class component (thay vì function component để viết state)

2. useState Hook

cú pháp: const [state, setState] = useState(initialValue);

3. Ví dụ về useState hook

gõ tên input => in ra giá trị

#33. Bài tập về State (part 1)

1. Lấy giá trị input khi submit button

- đặt state cho input
- submit button => lấy state đấy

Phân tích các bước làm:

1. đặt state
2. onchange => cập nhật state ấy
3. khi submit => lôi state ra sử dụng (vì state được lưu tại memory của component)

#34. Bài tập về State (part 2)

Logic render array, được đưa vào input component

Các bước làm:

tạo state quản lý array

khi submit btn => cập nhật array ấy

#35. Passing Function từ Cha sang Con (Parent to Child)

1. Khái niệm Parent/Child của React (cha/con)

Component nằm ngoài, bọc component khác, thì component đó là cha

2. Cách truyền function từ cha sang con

Để truyền function từ cha sang con, tương tự như truyền “giá trị của biến/variables”, chúng ta sử dụng props

Lưu ý có pháp viết code: chỉ truyền tên function, không có cặp dấu đóng mở ngoặc ()

Nếu truyền thêm dấu (), điều này đồng nghĩa chúng ta đang thực thi function đấy.

=> Từ component con, gọi function ấy, khi đấy mới dùng dấu ()

#36. Lift up State

- Lift-up state, giúp các component có thể nói chuyện với nhau (through qua component cha gần nhất)

Thực hành:

Đưa array cho component cha quản lý

truyền function update xuống cho child component

#37. Vòng đời của component

Component là "smart object". không "dump"/ngu ngốc như html.

Tương tự như "sinh vật", component có vòng đời của nó (life cycle)

Vòng đời gồm: sinh ra => lớn lên => die

Mounting (born) : được sinh ra -> chèn html vào giao diện (cây DOM)

Update: cập nhật giao diện sau khi đã có cây DOM

Unmounting (die): xóa html khỏi cây DOM

Component thể hiện vòng đời, thông qua props và state (vì state và props, giúp "hiển thị" dữ liệu của component)

=> props/state thay đổi => giao diện auto re-render

Ví dụ:

nút bấm button

#38. Về React Cơ Bản

Hiểu được props/state là gì ?

State: trạng thái của component. Cần lưu thông tin gì (variables, functions) của component, sử dụng state.

Về cách khai báo và sử dụng: **const [state, setState] = useState(initialValue)**

Props: properties - tài sản chuyển từ cha sang con. Có thể truyền (variables, functions) từ cha sang con để component con có "data" để sử dụng

Khái niệm render/re-render

State/props thay đổi => component re-render (vẽ lại giao diện mà không cần reload)

- Cập nhật state

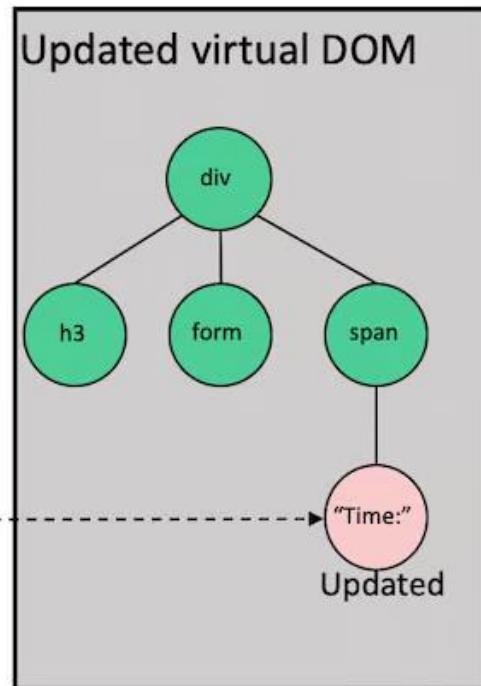
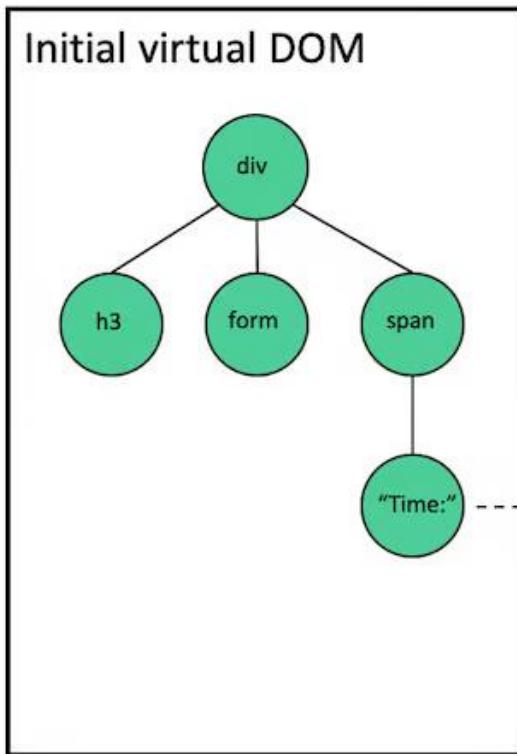
=> thông qua useState hook. **props của child component, thực chất là state của parent component**

React fiber ? (Virtual dom)

=> chỉ update những chỗ cần thiết

Batch update

<https://legacy.reactjs.org/docs/faq-internals.html>



Chapter 5: Setup Dự Án Backend

Giới thiệu về project thực hành và cách cài đặt backend (được cung cấp sẵn) cho dự án frontend React.

#39. Giới thiệu tổng quan về dự án thực hành

Frontend sẽ gồm 2 phần:

- React client (Vite), phụ trách quản lý CRUD của admin : localhost:5173
- React server (Next.js), normal user sử dụng : localhost:3000

Backend được cung cấp sẵn:

- Nestjs : localhost:8000
- database MongoDB, sử dụng online với mongodb atlas

Chuyện đánh bản quyền:

Điều khoản sử dụng dịch vụ khóa học của Hỏi Dân IT:

https://drive.google.com/file/d/1RR0ckOO_0Oyzh5kZ6ti0BPpOlySFn-NR/view

Tính trung thực

Cuộc chơi cân win-win

#40. Cài đặt Postman

- giới thiệu sơ lược về postman
- cài đặt local: <https://www.postman.com/downloads/>

- import collection => chưa cần test

Link download file collection: <https://drive.google.com/drive/folders/12mJcYGhHvkySgnOPBaopk0O44hWtxTm?usp=sharing>

#41. Cài đặt Mongodb Compass tại local

Link download: <https://www.mongodb.com/try/download/compass>

#42. Setup database MongoDB Atlas

- tạo tài khoản: <https://www.mongodb.com/cloud/atlas/register>

- test connection với MongoDB Atlas

- lưu ý về lỗi (thỉnh thoảng ko kết nối được) => tạo cái khác

#43. Kích hoạt dự án backend

Bước 1: Download & giải nén file **share-create-id.rar**

Link download:

<https://drive.google.com/drive/folders/12mJ-cY GhHvkySgnOPBaopk0044hWtxTm?usp=sharing>

Bước 2: Chạy dự án (chi tiết ghi tại file README.md)

- Chạy câu lệnh : **npm i** để cài đặt các thư viện cần thiết
- Update **file .env**, cập nhật thông tin
STUDENT_EMAIL : email của học viên Udemy
STUDENT_OS: hệ điều hành bạn đang sử dụng

Mỗi học viên khi mua khóa học, đều điền form này:

<https://forms.gle/A5s3wwFrNNDKQhnp7>

=> bạn điền email trong form trên là gì, thì bạn ghi vào vậy.

Lưu ý: điền chính xác email để đảm bảo quyền lợi

- Chạy dự án với câu lệnh: **npm start**

Bước 3: Lấy file kết quả ID

Nếu không có lỗi gì xảy ra, sẽ có message thông báo Done.

Kết quả được lưu tại file **/dist/data/result.txt**

Bước 4: Submit kết quả

Truy cập: <https://hoidanit-submit-license.vercel.app/license-sc>

Tại đây, bạn cần cung cấp các thông tin cần thiết, cũng như **file result.txt** (có được tại bước 3)

Nhấn submit

Bước 5: Nhận kết quả

Sau khi Submit kết quả ở trên, bạn chủ động inbox Facebook Hỏi Dân IT:

<https://www.facebook.com/askITwithERIC/>

Mình sẽ gửi kết quả qua email.

Thời gian phản hồi là 24h (từ thứ 2 tới thứ 6). 1 tới 3 ngày đối với thứ 7 & chủ nhật (ngày lễ/holiday)

#44. Run dự án backend

Bước 1: Download & giải nén file **share-backend-soundcloud.rar**

Bước 2: Chạy dự án

- Cài đặt thư viện cần thiết với câu lệnh: **npm install --omit=dev**
- Update file **.env**:
Cập nhật tham số cho database **MONGO_URL**
- Chạy dự án với câu lệnh: **npm start**

Bước 3: Kích hoạt bản quyền

Sử dụng kết quả nhận được email tại video #43, **cập nhật file result.txt**

Bước 4: Chạy lại dự án: **npm start**

Test dự án bằng cách:

Truy cập <http://localhost:8000/>

Test với postman APIs

Chapter 6: Routing với React (Client Side)

Điều hướng trang client với React Router, đồng thời tạo trang quản trị admin để quản lý dữ liệu trong dự án thực hành

#45. React Router

Tài liệu: <https://reactrouter.com/en/main>

1. Về cách đọc tài liệu

Chỉ có thể đọc được tài liệu cùng “main version”, không thể đọc tài liệu của chính xác version.

Có nghĩa là, có thể đọc được tài liệu chung của version 4.x, 5.x...

Nhưng không thể đọc điều tài liệu chính xác của 1 version cụ thể

=> giải pháp là nên đọc hiểu “cách dùng”, sau đấy áp dụng tương ứng cho dự án (vì công nghệ thay đổi theo thời gian)

2. React Router

What: được sử dụng để điều hướng trang mà không cần reload trang web.

How:

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

Cài đặt thư viện:

npm i --save-exact react-router-dom@6.15.0

#46. Tạo Route Users

Chưa cần design giao diện menu, quan tâm “hardcode “ route

Tài liệu:

<https://reactrouter.com/en/main/start/tutorial>

#47. Design table User

Các bước cần thực hiện:

- **Thư mục screens:** mỗi đường link url, là 1 screen
- **Thư mục components:** lưu trữ component được sử dụng ở các screens khác nhau (tăng tính tái sử dụng)

Yêu cầu:

Tạo component: manage.users.tsx

Render giao diện html table (hard code dữ liệu)

https://www.w3schools.com/html/tryit.asp?filename=tryhtml_table_intro

#48. Backend và API là gì

1. Tại sao cần backend ?

Tất cả các khóa học sử dụng “fake api” như json web server => vứt đi vì không có tính “thực tế cao”, chỉ phục vụ mục đích demo (testing)

Backend dùng miễn phí trên internet => không flexible

=> tự viết backend

Cần backend, vì backend có khả năng kết nối tới “database”, nơi lưu trữ “dữ liệu website”

2. Api là gì

Hiểu đơn giản, API là 1 đường link url.

Frontend gọi vào đường link url này để lấy dữ liệu

API được backend viết, Frontend sử dụng.

#49. Postman để test API

Yêu cầu:

- Đã cài đặt thành công Postman trên máy tính
- Đã có file collection và src backend (cũng như setup database)

Các bước thực hiện:

1. Run Backend
2. Run Postman (đã import collections)
3. Test APIs

Lưu ý khi chạy api có lỗi:

- Để biết được API đúng hay sai, dùng postman (với data được fill mẫu sẵn).

Nếu postman chạy được => frontend sai

- Backend không sai. Lỗi xảy ra, là frontend "truyền sai thông tin vào api".

Trường hợp cả postman và frontend (truyền đúng data), nhưng kết quả trả sai

=> APIs sai (các bạn report mình sẽ update)

#50. Gọi API với React

1. Công cụ hỗ trợ sẵn: Fetch

https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch

Fetch là công cụ hỗ trợ sẵn của javascript để gọi API phía client.

Với version node.js >= 18, có thể dùng fetch ở phía server

<https://nodejs.org/en/blog/announcements/v18-release-announce#fetch-experimental>

=> dùng fetch để thống nhất giữa client và server

2. Các thư viện hỗ trợ gọi APIs khác

- **axios**: (run client và server)

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

(các khóa học về React khác, ví dụ như khóa React Test Fresher, mình sử dụng axios)

<https://hoidanit.com.vn/khoa-hoc/giai-ma-trinh-do-bai-test-fresher-react.html?id=640beb7df7099c369b3bc695>

Tối ưu sẵn cache:

SWR: <https://www.npmjs.com/package/swr>

React Query: <https://tanstack.com/query/v5/docs/react/comparison>

3. Test API với Fetch

- VỚI client (fetch trong useEffect). Why ?

- VỚI server (fetch ở đâu cũng được)

- Test API login

- Test API get all users (hardcode bearer token ở header)

'Authorization': `Bearer \${token}`

#51. Mô hình stateless

- Stateless, là hình thức xác thực người dùng thông qua token (**access token/refresh token**).

Thông thường token sẽ được gửi kèm ở header request (hoặc cookies).

Xác thực người dùng gồm: người đấy là ai và người đấy có quyền làm gì (authentication và authorization)

Ví dụ: user là admin@gmail.com , có quyền CRUD user

- Token thường được gửi dưới định dạng JWT (**json web token**) đã được mã hóa và có tính bảo mật
- Mỗi lời gọi request được thực hiện từ Frontend tới Backend, đều cần đính kèm token.

Backend sẽ có cơ chế để "giải mã" token, từ đó, biết được người dùng nào đang thực hiện request (authentication), đồng thời có cho phép thực hiện hành động request hay không (authorization)

Frontend chẳng cần làm gì, ngoài việc:

1. login lấy token (lưu trữ lại, thông thường là localstorage và cookies)
2. Tất cả lời gọi request gửi lên Backend, đều cần gán thêm token trên.
3. Trường hợp token không hợp lệ (sai định dạng, hết hạn) => backend sẽ có thông báo lỗi trả về

#52. Hiển thị list User

- Đặt biến state

- Call API -> set state
(hard code token ở header)

- Render view
test sự thay đổi = cách dùng postman add new/delete a user

#53. Sử dụng key React hiệu quả

Tài liệu: <https://stackoverflow.com/questions/28329382/understanding-unique-keys-for-array-children-in-react-js/43892905#43892905>

1. Tại sao React cần key

key là cách react "xác định/định danh" phần tử HTML trên màn hình (tương tự ID)

Component sẽ dựa vào key để tối ưu hóa hiệu năng khi:

- Edit
- Delete
- Add new

=> chỉ có phần tử nào được edit/delete/add new thì thay đổi nơi đó. không render lại toàn bộ component (chi phí paint đắt, dễ gây giật/lag)

2. Chuyện key = index của array

<https://github.com/facebook/react/issues/1342#issuecomment-39230939>

Mặc định, nếu không đặt key, React "tự động" lấy key = index

<https://github.com/facebook/react/issues/1342#issuecomment-39274004>

Thông thường, nếu chúng ta không modify (thêm/sửa/xóa phần tử, filter, sort... của array => thay đổi thứ tự -index).

=> đặt key = index vẫn hoạt động ok

Trường hợp đặt key = index gây ra bugs: (tương tự với random key/uuid cho key)

Nguyên tắc của key:

1. Unique (duy nhất): các key cần khác nhau (không giống nhau) => id
2. static (tĩnh): key không được thay đổi giữa các lần render. vì key thay đổi => react vẽ lại phần tử đấy (replace)

Ví dụ về bad code:

ví dụ **arr = ['fresher', 'junior', 'middle', 'senior'];**

```
<ul>
{
  arr.map((item, index) => {
    return <li key={index}> {item} </li>
  })
}
</ul>
// kết quả:
<ul>
  <li key=1>fresher</li>
  <li key=2>junior</li>
  <li key=3>middle</li>
  <li key=4>senior</li>
</ul>
```

Nếu thao tác "modify" (sửa đổi) mảng trên, ví dụ: **xóa phần tử 'fresher'**

```
=> new arr = ['junior', 'middle', 'senior' ];
//kết quả
<ul>
  <li key=1>junior</li>
  <li key=2>middle</li>
  <li key=3>senior</li>
</ul>
```

=> tất cả phần tử đều thay đổi key => react cần vẽ lại toàn bộ giao diện @@

//Part 2:

Cách sử dụng key hiệu quả

1. Sử dụng id (backend cung cấp)

2. Nếu không có id, cần convert dữ liệu để tạo ra id => rồi map

không dùng uuid, hay random number "bên trong vòng map"

```
{  
arr.map((item, index) => {  
    return <li key={uuid()}> {item} </li>  
})  
}
```

=> dùng như trên, mỗi lần render, tất cả các phần tử đều replace (do bị thay đổi)

Chapter 7: Hướng dẫn sử dụng Chrome Dev Tool (Bổ Trợ)

Một chương đặc biệt hướng dẫn kỹ năng sử dụng Devtools, phục vụ đắc lực cho quá trình debug code. Chương này đặc biệt, vì rất nhiều bạn không có kỹ năng debug này.

#54. Thành phần của dev tool

- Tab Elements: Xem giao diện html, code css trực tiếp (hiểu đơn giản là cây DOM)
- Console: check thông tin code: info, error, warning
error => fix khi code die, không đồng nghĩa là error => fix
warning => fix được thì càng tốt
- sources: => dùng để đặt breakpoint (quá trình debug, xem code chạy)
 - Network: dùng để check api giữa frontend và backend
Lưu ý về red button (record network)
-> chọn options fetch/xhr để filter data
 - Application: check local storage, session storage, cookies, indexedDB ... (data lưu tại browser)

#55. Kỹ thuật check api

Sử dụng tab network để kiểm tra api

Nhớ đã chọn records network

Chọn api, đọc thông tin của api, bao gồm:

Tab headers => check url , method, thông tin ở headers...

Tab payload (nếu có): thông tin đính kèm api

Preview (xem phản hồi dưới dạng object)

Response: xem phản hồi dưới dạng text

#56. Nguyên tắc khi gọi API không có kết quả

- check postman (với data mặc định) xem có kết quả hay ko
- f12 check tab network. check url, method, data... đã chính xác chưa

- Minh họa trường hợp truyền sai data

Chapter 8: Module Users

Thực hành bài tập quản lý người dùng hệ thống qua ví dụ CRUD (create, read, update, delete)

#57. Tích hợp antd

Tài liệu: <https://ant.design/>

Giới thiệu antd, bootstrap, MUI

Cài đặt antd

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

npm i --save-exact antd@5.8.4

Hướng dẫn cách đọc tài liệu: vào trang chủ của antd, chọn component => copy paste thôi :v

#58. Design giao diện Admin

Part 1:

Tài liệu:

<https://reactrouter.com/en/main/start/tutorial#nested-routes>

<https://ant.design/components/menu>

Part 2:

Tài liệu:

<https://reactrouter.com/en/main/components/link>

#59. CSS với React

Phân loại CSS (mở trang tài liệu nextjs): <https://nextjs.org/docs/app/building-your-application/styling>

Why not tailwind ?

=> tailwind thiên nhiều về code HTML thuần => chưa đóng gói component

#60. Tích hợp SASS (SCSS)

Cài đặt sass

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

npm i --save-exact sass@1.66.1

Lưu ý: viết code modules

#61. Antd Table

Tài liệu:

<https://ant.design/components/table>

Gồm 2 thành phần:

dataSource (list object lấy từ backend)

columns (mapping theo dataIndex)

#62. Hiển thị danh sách Users

//Todo

#63. Antd Modal

Tài liệu:

<https://ant.design/components/modal>

#64. Design Modal Add New

Design input

<https://ant.design/components/input>

Submit lấy được data khi submit

#65. API Add New User

Lưu ý:

Cần truyền jwt ở header cho api

Tại giao diện frontend, làm tương tự như api login (gán bearer token ở header request)

POST <http://localhost:8000/api/v1/users>

Truyền body của request các tham số:

name, email, password, age, gender, address, role

#66. Bài tập Add New User

Sau khi add new user => fetch lại danh sách

#67. Bài tập Design Modal Update User

Khi nhấn user => tự fill thông tin input

Part 1: tách code

Part 2: design update modal => fill thông tin

Part 3 : complete

#68. Bài tập Update User

Gọi api update

Submit => fetch lại data

#69. Design Delete User

Popconfirm

<https://ant.design/components/popconfirm>

Api delete

#70. Bài tập Delete User

Popconfirm

Api delete

#71.1 Pagination

1. What/Why ?

Pagination (phân trang) là việc chia nhỏ dữ liệu theo từng page (trang).

Ví dụ = cách search google,

kết quả tìm kiếm sẽ được chia ra làm page 1, page 2, ..., page n

Why ?

Cần phân trang dữ liệu vì:

- Tại 1 thời điểm, user chỉ có thể "nhìn thấy 1 lượng nhỏ data, mà không nhìn thấy tất cả", ví dụ 10, 20 kết quả tìm kiếm

- Nếu load tất cả data, càng nhiều data, tốc độ load sẽ càng chậm (hình dung việc download file, file càng nặng, thời gian chờ đợi càng lâu, và ngược lại)

=> việc phân trang giúp tăng trải nghiệm của người dùng, tức là "load dữ liệu đủ dùng", không thừa thãi.

2. Nguyên tắc khi phân trang

- Trong thực tế, việc phân trang kết quả, backend làm, frontend chỉ sử dụng (cung cấp thông tin cần thiết cho backend)

- Tuy nhiên, frontend cần hiểu "nguyên tắc phân trang", như vậy phối hợp với backend mới dễ.

3. Cách phân trang

thông thường, frontend sẽ truyền lên 2 tham số:

- số lượng bản ghi tối đa cần lấy (max/limit) là bao nhiêu => pageSize
- trang muốn lấy (page)

tham khảo: https://www.w3schools.com/php/php_mysql_select_limit.asp

ví dụ:

trong database có 101 bản ghi: R1, R2, ..., R100, R101.

Nếu phân trang, một trang hiển thị 10 kết quả
=> số trang tối đa là $101/10$ (làm tròn) = 11 trang

trang 1: R1, R2, ..., R10

trang 2: R11, R12, ..., R20

...

trang 10: R91, R92..., R100

trang 11: R101 (chứa 1 kết quả)

=> frontend chỉ cần truyền lên pageSize (limit) và trang muốn lấy kết quả.

ví dụ : pageSize = 10, page = 2 => backend sẽ cần trả ra kết quả: R11, R12, ..., R20
(10 bản ghi tương ứng)

#71.2 Antd Pagination

Tài liệu:

<https://ant.design/components/pagination#components-pagination-demo-total>

#71.3 Update Token

1. Local Storage là gì ?

2.

//todo. sử dụng access token thông qua localStorage
vào app => gọi api login => set localstorage

các page khác, lấy token từ localStorage

#72. Bài tập Paginate User

//todo

Chapter 9: Tối ưu hóa hiệu năng Form React

Một tiêu chuẩn "ngầm định" khi sử dụng form với React để xử lý nhiều data, đấy chính là sử dụng uncontrolled component, thay vì state.

#73. Controlled Component vs Uncontrolled Component

1. Khái niệm

- **Controlled component:** "Kiểm soát" component thông qua state/props của React.
Dùng state/props để khiến component re-render

ví dụ: <input/> có gán state và onChange (cách đã làm khi create/update user)

- **Uncontrolled component:** "Không kiểm soát" component (không dùng state/props)

2. Ví dụ

- + khi "controlled" quá nhiều component, và các component này "re-render" cùng lúc
=> hiện tượng giật/lag

(thông thường sẽ xảy ra khi có từ hàng trăm component re-render cùng thời điểm)

//todo: minh họa với web về tính năng tạo mục lục

- + Uncontrolled Component

- Thao tác với html thuần => không khiến component re-render nên không bị giật lag

3. Khi nào sử dụng

- Controlled component:

- + muốn "kiểm soát" component
- + dữ liệu ít/hoặc re-render ít

- Uncontrolled component

- + Sử dụng với form có nhiều dữ liệu (win => standard)

#74. Thư viện hỗ trợ Uncontrolled Component

- Khi không dùng react (state/props), câu hỏi đặt ra là làm sao để có thể lấy được dữ liệu của component ?

=> thư viện ra đời.

1. Một vài thư viện nổi tiếng

- React Hook form

<https://react-hook-form.com/>

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

- Formik

<https://formik.org/docs/examples/basic>

2. Antd Form

- Được base dựa trên formik

- Giao diện thân thiện, dễ dùng hơn

- handle validate :v

<https://ant.design/components/form>

=> sử dụng form method để có thể get/set dữ liệu của form

#75. Create User (Uncontrolled Component)

//todo

#76. Update User (Uncontrolled Component)

//todo

Chapter 10: React Hiện Đại với Next.JS

React hiện đại, và tương lai của React, chính là Framework Next.JS

#77. Vấn đề tồn đọng với cách viết React cũ (CSR)

1. Khái niệm SPA

SPA: single page application

CSR: client side rendering

2. Nhược điểm của CSR

- Tốc độ load trang chậm (với máy cấu hình yếu)

3. SEO

#78. Nextjs là gì

1. What ?

Next.js là framework javascript, tích hợp sẵn React, dùng để build website -framework: có nghĩa rằng, nó đã cho sẵn bộ khung và làm sẵn nhiều tính năng, chúng ta chỉ dùng thôi

- tích hợp sẵn react => code UI sẽ sướng hơn

2. Why Nextjs ?

- **Nextjs** là framework => được làm sẵn nhiều cái (ví dụ như **router**, **images**, ...)

- tích hợp React (ở phía server). có nghĩa là, vẫn code được React, tốc độ website nhanh hơn (server side rendering)

- **Hỗ trợ SEO tốt (nếu cần SEO)**

=> Nextjs mang tới sự flexible (code React ở Frontend/Backend), đồng thời hỗ trợ SEO tốt

#79. Nextjs làm frontend hay backend ?

1. Nextjs làm frontend hay backend ?

- **Nextjs chỉ nên làm frontend (mặc dù có thể code logic backend trong dự án Nextjs).**

Lý do:

+ Cấu trúc của nextjs, thiên về frontend. bạn muốn code BE => **tự đi mà code**

khi build thêm BE => khôi lượng code x2

=> hiệu năng giảm, trong khi công sức maintain x2

+ Yêu cầu về an toàn thông tin

2. Phân biệt nextjs và nestjs

- **Nextjs** là framework tích hợp React, chuyên làm frontend

- **Nestjs** là framework tích hợp typescript (mặc định), dùng để làm backend với tính mở rộng cao

Nếu nextjs làm luôn backend, tại sao nestjs tồn tại :v

#80. Setup dự án với Nextjs (tích hợp MUI)

Tài liệu:

<https://mui.com/material-ui/getting-started/example-projects/>

<https://github.com/mui/material-ui/tree/master/examples/material-ui-nextjs-ts>

1. Run Project

Link source code: <https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-mui-ts-starter>

Version node.js: 18.17.0

2. Tự tạo project

Lưu ý: trong khóa học này, bạn vui lòng thực hiện bước 1 (clone project từ source code cung cấp), không tự setup dự án. Có bugs, mình không có trách nhiệm fix bugs hộ đâu :v

Chúng ta chỉ thực hiện bước này, khi và chỉ khi, chúng ta biết chúng ta đang làm gì (và đã có khả năng tự fix bug).

Có 3 cách:

- Check tài liệu của Nextjs:
<https://nextjs.org/docs/getting-started/installation>
- Check example của Nextjs:
<https://vercel.com/templates/next.js>
- Check example của MUI:
<https://mui.com/material-ui/getting-started/example-projects/>
<https://mui.com/material-ui/guides/server-rendering/#material-ui-on-the-server>

Download thư mục của github: <https://download-directory.github.io/>

#81. Cấu trúc dự án Next.js

1. Tại sao lần đầu vào website thường chậm ?

Quá trình chạy dự án:

- Nextjs sẽ cần dịch code typescript thành javascript
- Tạo thư mục **.next (code đã được dịch)**
- Lưu cache vào memory

Trong lần đầu tiên vào website (chế độ development/localhost), nextjs cần dịch code và tạo cache => lần đầu thường lâu

Từ lần thứ 2 trở đi sẽ nhanh hơn (do tái sử dụng cache)

2. Cấu trúc dự án

Công nghệ sử dụng (môi trường node.js v.18 và typescript v.5)

- React 18
- Next.js 13 (App router)
- MUI 5

Cấu trúc dự án tương tự như dự án React sử dụng với Vite.

#82.1 React children

Tài liệu: <https://react.dev/learn/passing-props-to-a-component#passing-jsx-as-children>

Ở đây, là đề cập cách sử dụng children as a props.

#82.2 React Fragment

Tài liệu:

<https://react.dev/reference/react/Fragment>

Bài toán: nguyên tắc render 1 block với JSX

Ví dụ:

```
render (  
  <div> ... </div> // ok  
)
```

```
render (  
  <div classname="1"> ... </div>  
  <div classname="2"> ... </div> // báo lỗi  
)
```

Giải pháp thông thường, là dùng 1 phần tử HTML nào đấy bọc ngoài, ví dụ <div>

```
render (  
  <div classname="3"> //ok  
    <div classname="1"> ... </div>  
    <div classname="2"> ... </div>  
  </div>  
)
```

1. Vấn đề gặp phải

- Sử dụng phần tử HTML bọc ngoài sẽ phá vỡ cấu trúc HTML ban đầu
- HTML thay đổi => ảnh hưởng tới CSS

2. Cách giải quyết với Fragment

Fragment sinh ra để giải quyết vấn đề trên, mà không làm thay đổi cấu trúc của HTML

before:

```
render ()  
<div classname="3"> //ok  
    <div classname="1"> ... </div>  
    <div classname="2"> ... </div>  
</div>  
)
```

after:

```
render ()  
<> //ok  
    <div classname="1"> ... </div>  
    <div classname="2"> ... </div>  
</>  
)
```

Fragment được sử dụng với `<> </>` hoặc `<Fragment> </Fragment>`

Chỉ dùng `<Fragment>` khi cần render key

<https://react.dev/reference/react/Fragment#rendering-a-list-of-fragments>

```
return posts.map(post =>  
    <Fragment key={post.id}>  
        <PostTitle title={post.title} />  
        <PostBody body={post.body} />  
    </Fragment>  
)
```

#83. Cách đọc tài liệu của Nextjs

Trang tài liệu của Next.js:

<https://nextjs.org/docs>

Lưu ý về version của Nextjs:

Với Nextjs 13, breaking change đã xảy ra.

Tất cả kiến thức của next 12 trở về trước, sẽ không áp dụng với next 13 (và các version lớn hơn)

- Sử dụng app router (thay vì page router)

#84. Chuyện Next.js ngốn RAM

Yêu cầu tối thiểu cần 8GB RAM

không nên sử dụng react-icons (nhìn số lượng modules compile)

Chuyện ngốn RAM là tốt hay xấu:

chỉ xảy ra tại môi trường development

=> why (chạy build rất ok), vì load in-memory

=> không ảnh hưởng tới môi trường production, chỉ cần lưu ý trong quá trình code để có trải nghiệm tốt hơn.

Chapter 11: Next Routing

Điều hướng trang với framework Next.js

#85. Tạo base giao diện

Lưu ý: trước khi bắt đầu, clear giao diện (xóa tất cả)

- chỉ để lại homepage

#86. Routing với nextjs

- Để điều hướng trang với Next.js, chúng ta "không cài đặt thêm thư viện", ví dụ như react-router (cách làm với react Vite)

Tính năng điều hướng trang đã được tích hợp sẵn trong framework này.

- Nextjs sử dụng "tên thư mục" (folder) để định nghĩa route

1. Khai báo route

- Với Nextjs < 13 (version cũ hơn 13), chúng ta có thư mục "**pages**"
=> khai báo route ở trong folder "pages"

- **Với Nextjs 13 (hoặc lớn hơn), sử dụng thư mục "app" (đây là cấu hình mặc định)**

- Nextjs sử dụng "tên thư mục" để định nghĩa route
=> tạo thêm folder bên trong thư mục "app"

2. File conventions

layout: share layout giữa các page con.

page: định nghĩa giao diện cho page

tên file là .js, .jsx hoặc .tsx

3. Minh họa

//todo: download this image:

<https://nextjs.org/docs/app/building-your-application/routing#colocation>

#87. Định nghĩa route

- **homepage** : hiển thị tracks
- **profile**: thông tin cá nhân
- **playlist**
- ko cần làm login/register => sẽ làm ở module auth login
- detail track (làm sau)

#88. Phân biệt cách code UI với React

Có 2 cách thường dùng để code UI (user interface/giao diện website) với React, đó là "code chạy từ a tới z", hoặc là sử dụng thư viện đã design sẵn component

1. Code chạy (code thuần)

Code chạy, đồng nghĩa với việc bạn làm từ a tới z mà không sử dụng thư viện bên ngoài **ưu điểm:**

- **kiểm soát mọi thứ** và hiểu từng dòng code, đơn giản là do bạn viết từ a tới z mà

nhược điểm:

- UI sẽ xấu nếu như level của bạn thấp
- Tốn thời gian, vì đôi khi sẽ rơi vào tình trạng "**reinvent the wheel**", đi làm lại tính năng mà người khác đã làm "hoàn hảo" :v
- chưa tự duy theo thiên hướng "**component**" (reuse component), đang thiên nhiều về HTML
- đôi khi bạn làm từ a tới z, cơ mà 1 tháng sau, 1 năm sau đọc lại code đấy, chưa chắc bạn đã hiểu đống shit (code) ấy là gì :v

Ví dụ về cách code chạy:

- Sử dụng CSS/SASS, SCSS/LESS:
 - + Cách viết này là định nghĩa component của React, sau đấy style CSS thuần túy
- Sử dụng thư viện/Framework cho CSS, ví dụ **bootstrap, tailwind**
 - + Cách viết này là định nghĩa CSS ngay trong HTML, tương tự inline CSS. Điểm khác biệt là định nghĩa classname

Ví dụ:

```
<div class="h-screen w-full overflow-hidden flex flex-nowrap text-center" id="slider"/>
```

Bonus:

Tại sao Tailwind lại sử dụng với Next.js (các series ngoài kia):

- Tailwind, thiên về HTML (khi bổ sung sự mạnh mẽ cho code CSS)
- Nextjs là SSR (server side rendering), hỗ trợ rất tốt cho HTML
- Tailwind setup đơn giản với Nextjs

2. Sử dụng UI Component

- Dùng UI Component được cung cấp bởi các thư viện để sử dụng.
Nhiệm vụ của chúng ta là import Component và sử dụng theo ý muốn.

- Các thư viện UI phổ biến hay dùng với React, có thể kể tới như:

+ React Bootstrap (nếu bạn đã từng code bootstrap thuần):

<https://react-bootstrap.netlify.app/docs/getting-started/introduction>

+ Ant Design: <https://ant.design/components/overview>

+ MUI

Ưu điểm:

- UI đẹp, chuyên nghiệp (professional)
- Tiết kiệm được thời gian design UI (cho dù bạn level thấp)
- Tư duy theo hướng component (tăng khả năng tái sử dụng)
- Hỗ trợ responsive layout

Nhược điểm:

- Bạn cần đọc tài liệu của thư viện để biết cách sử dụng
- Bạn cần đọc tài liệu của thư viện để biết cách customize theo ý muốn
- Tốn thêm công sức setup thư viện vào các framework phức tạp như Next.js

3. Why MUI với Next.js

- Không dùng Bootstrap vì giao diện "đơn điệu", không phong phú, đồng thời, mình đã sử dụng Bootstrap trong khóa React Ultimate
- Không dùng Antd, vì antd không thích hợp cho giao diện client và hỗ trợ không tốt với Nextjs.
đồng thời, Antd đã được sử dụng với React Vite

bạn vẫn có thể sử dụng Antd với Nextjs, cơ mà nên cân nhắc một vài trade off sau:

- + khối lượng bundle của Antd tương đối lớn (do tích hợp sẵn nhiều thư viện UI)
- + Việc tạo cache để tối ưu hiệu năng từ server với Nextjs cần config chính xác

+ các issue/community của Antd đa phần là tiếng Trung Quốc

- Sử dụng MUI vì:

+ học thêm thư viện UI mới mà nhiều công ty sử dụng

+ giao diện UI đẹp (mặc dù chưa đầy đủ bằng Antd)

+ hỗ trợ tốt cho Nextjs (tốc độ hỗ trợ nhanh hơn Antd)

+ các issue/community của MUI là tiếng Anh

#89. MUI là gì

Tài liệu:

<https://mui.com/>

1. What

- MUI là 1 thư viện cho phép sử dụng các component được viết sẵn dành cho React
- Phong cách design của MUI bị ảnh hưởng bởi **Google's Material Design** (giao diện sam same)

- Phân loại MUI:

Chúng ta sử dụng Material UI để kế thừa lại các component đã viết sẵn:

<https://mui.com/material-ui/getting-started/>

2. Why ?

- MUI hỗ trợ **cache** với nextjs
- **styled component** (phong cách code css với MUI), tương tự như code **React Native**
- **MUI so với antd, bootstrap và tailwind ?**
 - + MUI có ít tính năng hơn antd, tuy nhiên hỗ trợ Nextjs tốt hơn antd
 - + MUI đẹp hơn bootstrap
 - + MUI hỗ trợ sẵn component, trong khi với tailwind bạn cần code thuần.

#90. Cách sử dụng MUI component

1. Tài liệu

Trong khóa học này, chúng ta sử dụng Material Design:

<https://mui.com/material-ui/getting-started/>

2. Phong cách code của MUI

- Bạn có thể code CSS với Javascript (**styled component**)

=> tương tự React Native

<https://emotion.sh/docs/introduction>

<https://styled-components.com/>

why styled components:

<https://styled-components.com/docs/basics#motivation>

Chapter 12: Module HomePage

Design trang chủ clone soundcloud

#91. Bài tập Design App Bar

Part 1: Phân tích yêu cầu

Link design: <https://github.com/harypham/soundcloud-idea/wiki/2.Design-Frontend#root-hierarchy>

Tài liệu:

<https://mui.com/material-ui/react-app-bar/>

<https://mui.com/material-ui/react-app-bar/#app-bar-with-a-primary-search-field>

<https://mui.com/material-ui/react-container/>



Part 2: Chữa bài tập

Link source code #91.2:

<https://drive.google.com/file/d/1-ThN7qp8-rdDeggFaPDalZ-TewC4ER92/view?usp=sharing>

Part 3: Chữa bài tập

Link source code #91.3:

https://drive.google.com/file/d/1DytjRJAa8VHNKzD_qpqZDeO-7FTB1SrB/view?usp=sharing

Tài liệu:

<https://mui.com/material-ui/react-avatar/>

<https://mui.com/material-ui/react-menu/>

<https://mui.com/material-ui/react-menu/#account-menu>

#92. Navigating

Tài liệu: <https://nextjs.org/docs/app/building-your-application/routing/linking-and-navigating>

Yêu cầu:

- Nhấn vào logo => redirect “/”
- Nhấn vào Playlist => redirect “/playlist”
- Nhấn vào Likes => redirect “/like”
- Nhấn vào Profile => redirect “profile”

#93. Share Layout

//todo

#94. Server Component

Tài liệu: <https://nextjs.org/docs/app/building-your-application/rendering/server-components>

Mặc định, tất cả “component” của Nextjs, là Server Component

Server Component chỉ đơn thuần là HTML (không có sự tương tác của Javascript)
=> không dùng các function xử lý sự kiện hay các hook của React

Lợi ích của server component:

- Tốc độ load page nhanh
- Cache
- Security
- SEO

Server component có thể bọc “Client component”

#95. Client Component

Tài liệu: <https://nextjs.org/docs/app/building-your-application/rendering/client-components>

Cần có sự tương tác => sử dụng client component

Ví dụ: useState, useEffect...

//todo

#96. Use Client

Nhắc lại:

Server component: render 100% trên server, tạo ra file HTML gửi về cho client.
Client chỉ việc hiển thị mà không cần làm gì khác.
=> hiểu đơn giản là SSR (server side rendering), so sánh với CSR (client side rendering) khi dùng với React thuần (Vite)

Client component: render ở phía client liệu có chính xác ?

- Giúp có sự tương tác với website
- Tương tác với browser API, ví dụ localStorage, windows...
- Để sử dụng client component, cần sử dụng "use-client"

1. Use-client

Use client, không có nghĩa là 100% component sẽ được render ở phía client (như client side rendering)

Nhiệm vụ của Nextjs là **pre-render** trước khi gửi về cho client.

Mô hình khi sử dụng use-client:

Phase 1: pre-render

Nextjs sẽ tạo trước file html mà không có sự tương tác của HTML, sau đây gửi về client (như vậy được ½ quá trình)

Phase 2: hydrate

Client nhận về file HTML đã được render trước từ Nextjs, sau đây dùng React để thêm phần tương tác Javascript

#97. Bài Tập Design Main Content

Part 1: Giới thiệu

Tài liệu: <https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/design-wiki/-/wikis/2.Design-Frontend#2-main-content>

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

<https://ant.design/components/carousel>

Part 2: Setup

giới thiệu về ký hiệu của npm (datatype) DT/TS

ví dụ: nextjs (TS)

react (DT)

npm i --save-exact react-slick@0.29.0 slick-carousel@1.8.1

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

<https://react-slick.neostack.com/>

Part 3: Design Main content

Part 4:

Source code video này:

https://drive.google.com/file/d/1IKZgN7Rn48VGlUMrS0j_cHn_i9-JKIBV/view?usp=sharing

<https://mui.com/material-ui/react-box/>

<https://mui.com/material-ui/react-divider/>

#98. Design Footer

Part 1: setup

Tài liệu:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/fe-design-wiki/-/wikis/2.Design-Frontend#3-footer-track-player>

Yêu cầu:

audio player, stick at the bottom

Setup thư viện:

<https://www.npmjs.com/package/react-h5-audio-player>

npm i --save-exact react-h5-audio-player@3.8.6

Part 2: Gợi ý

Sử dụng các component:

<AudioPlayer/> của thư viện để có thể play audio

Tạo nhanh Appbar stick to bottom:

<https://mui.com/material-ui/react-app-bar/#bottom-app-bar>

Tạo layout với Container:

<https://mui.com/material-ui/react-container/>

Part 3: Review

Source code video này:

https://drive.google.com/file/d/1SGxaaN1_eZBSKN1aAE5WxND_vq7rVsHW/view?usp=sharing

https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/dev/src/utils/customHook.ts?ref_type=heads

#99. Nextjs Environment Variables

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/configuring/environment-variables>

Mặc định, nextjs đã support .env => không cần cài đặt thêm thư viện nào khác

Tạo file .env ở root folder

Với môi trường development => .env.local / .env.development

môi trường production => .env.production

=> không quan tâm về môi trường => .env

Lưu ý về môi trường browser và server:

<https://nextjs.org/docs/app/building-your-application/configuring/environment-variables#bundling-environment-variables-for-the-browser>

=> những tham số nào chạy ở browser => thêm tiền tố NEXT_PUBLIC_

2. Áp dụng

Tạo .env cho url backend

#100. Fetch music from backend

-- sử dụng .env để lưu url backend

<http://localhost:8000/tracks/hoidanit.mp3>

#101. Fetch data với Nextjs

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/data-fetching/patterns>

1. Hiểu về Next.js

Node.js là môi trường thực thi javascript.

javascript có thể chạy tại client hoặc server

client: browser

server: các máy chủ

Nextjs = code javascript chạy tại server + code js gửi về cho client

2. Fetch Data

Whenever possible, we recommend fetching data on the server

=> Nếu có thể, ưu tiên fetch data trên server

Với Nextjs, có 2 cách fetch data:

1. Fetch data trên server (logic fetch data chạy tại server) (Server side rendering)

=> sử dụng được cách này, vì server có khả năng kết nối tới database

2. Fetch data tại browser (sử dụng tại **useEffect**, mô hình Client side rendering)

Tại client, không thể viết code như cách fetch trên server, vì môi trường chạy code

là browser

=> **bắt buộc viết trong hàm useEffect (khi DOM đã sẵn sàng)**

#102. Fetch Data HomePage

Lưu ý: chưa sử dụng Next Image

POST: <http://localhost:8000/api/v1/tracks/top>

(đã sắp xếp theo countPlay)

Truyền body:

category: string; //CHILL/PARTY/WORKOUT

limit: number

#103. Fetch Wrapper

một vài thư viện tham khảo:

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

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

Tài liệu:

https://dev.to/ajmal_hasan/api-wrappers-using-axios-fetch-55dl

npm i --save-exact query-string@8.1.0

//Part 1: sử dụng javascript

Source code mẫu: https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/dev/src/utils/old.api.js?ref_type=heads

//Part 2: typescript

tạo data type: d.ts (setup global)

Source code mẫu: https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/dev/src/utils/api.ts?ref_type=heads

https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/dev/src/utils/backend.d.ts?ref_type=heads

Source code video này:

https://drive.google.com/file/d/1uCAEWQItepbfGMhAVhQwekVu9YF_ANh-/view?usp=sharing

#104. Hoàn thiện HomePage

//todo: add source code

POST: <http://localhost:8000/api/v1/tracks/top>

(đã sắp xếp theo countPlay)

Truyền body:

category: string; //CHILL/PARTY/WORKOUT

limit: number

//Part 2:

Link images: <http://localhost:8000/images/chill1.png>

Source code video này:

<https://drive.google.com/file/d/1dz2nsR8v3AvAl9KtOkS5a7ZP88hWSrAk/view?usp=sharing>

Chapter 13: Module Wavesurfer

Tạo wave sound tương tự như Soundcloud bằng cách tích hợp thư viện javascript với React

#105. Dynamic Route

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/routing/dynamic-routes>

=> tạo route track/[slug]

/tracks/id

Dùng id vì id là unique

Lưu ý: chưa quan tâm về SEO với URL

#106. Search Params

/tracks/id?audio=abc.mp3

Tài liệu

<https://nextjs.org/docs/app/api-reference/functions/use-search-params>

#107. Giới thiệu về Wavesurfer

Lấy keyword từ soundcloud

Tài liệu:

<https://wavesurfer-js.org/>

<https://www.npmjs.com/package/wavesurfer.js>

Cài đặt:

npm i --save-exact wavesurfer.js@7.3.1

1. Tại sao không dùng Wrapper, lại dùng trực tiếp thư viện JS

Thư viện đã tích hợp wavesurfer dưới dạng component:

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

Không dùng wrapper vì:

1. library wrapper thực chất cũng sử dụng wavesurfer.js, tương tự như cách chúng ta viết fetch wrapper

2. library wrapper chưa sử dụng wavesurfer.js version mới nhất

=> đôi khi đọc tài liệu của wavesurfer.js sẽ không áp dụng được

<https://github.com/ShiiRochi/wavesurfer-react/blob/master/package.json>

3. Học trực tiếp wavesurfer.js, là cách chúng ta tích hợp 1 thư viện javascript vào React

#108. Next.JS Public Folder

Tài liệu: <https://nextjs.org/docs/app/building-your-application/optimizing/static-assets>

Có thể truy cập file từ Nextjs, thông qua thư mục Nextjs

Minh họa với Image và Audio

#109. Wavesurfer Basic

Tài liệu:

<https://wavesurfer-js.org/examples/?basic.js>

Lưu ý: sử dụng basic ID, chưa sử dụng Ref

Khi chạy tại chế độ development, hàm useEffect sẽ được chạy 2 lần (React cố tình làm vậy để tối ưu hóa/phát hiện bugs)

Chạy tại chế độ production, hàm useEffect chỉ chạy đúng 1 lần, nên là anh/em yên tâm nhé

#110. React Ref

- component: wave.track.tsx

Vấn đề tồn đọng với ID ?

=> không thể có cùng 2 (hoặc nhiều hơn) component trên cùng màn hình do trùng ID

1. Ref là gì

Tài liệu:

<https://react.dev/learn/referencing-values-with-refs>

Ref hay được dùng nhất, để có thể "thay đổi" HTML (DOM)

Nếu bạn dùng JQuery để thay đổi HTML, với React -> sử dụng Ref

Ref.current => trả ra giá trị nó đang "hold"

2. Update component với Ref

//todo

#111. Fetch track from Remote

- dynamic url from backend

Part 1: Lỗi CORS khi gọi API ?

Tài liệu: <https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>

1. CORS là gì ?

CORS === Cross-Origin Resource Sharing

CORS là cơ chế "mặc định" của trình duyệt web (browser) khi gọi/truy cập thông tin khác tên miền (domain)

Ví dụ: từ google.com gọi sang facebook.com -> browser chặn CORS
từ localhost:3000 -> gọi localhost:8000 -> chặn CORS

CORS chỉ xảy ra tại browser, giúp duyệt/lướt web trở nên an toàn hơn.

Why ?

Với browser, khi bạn truy cập/gọi tới 1 website bất kỳ, browser sẽ tự động gửi cookies kèm theo

Browser có cookies, nên nếu không chặn CORS, sẽ tìm ẩn nguy cơ ăn trộm cookies

2. Cách khắc phục CORS ?

- Nếu bạn sử dụng cơ chế **client (browser)** -> **gọi Server để lấy dữ liệu**, fix lỗi CORS trên server

- Nếu bạn không code backend, có thể fix = cách disable cors của browser (NOT recommended)

- Sử dụng same domain, hoặc server gọi tới server

Mô hình:

Nextjs (client) -> gọi tới **Nextjs (server)** : không bị CORS do cùng domain (localhost:3000)

Nextjs (server) -> gọi tới **Backend**

Nextjs (server) trả kết quả cho Nextjs (client)

Part 2: Nextjs Route handler

Tài liệu: <https://nextjs.org/docs/app/building-your-application/routing/route-handlers>

- Viết API với Nextjs

Part 3: Complete

- sử dụng next route (viết api trên server) => tránh cors
- component dynamic.url.track.tsx

Source code video này:

<https://drive.google.com/file/d/1DAPhyBerJdQTtlWRvIMF-rf4LWn-e6T/view?usp=sharing>

#112. Tạo wavesurfer hook

Tài liệu:

<https://wavesurfer-js.org/examples/?react.js>

1. Tại sao cần tạo wavesurfer Hook

- Mục đích: truy cập tới instance của wavesurfer

Nhờ có instance (đối tượng), chúng ta có thể thao tác được các functions mà thư viện cung cấp mọi lúc, mọi nơi

Source code video này:

https://drive.google.com/file/d/1e63Cq47v_Ncww4hKkd1S7HibiCDp53kl/view?usp=sharing

#113. Sử dụng useMemo hook

Tài liệu:

https://www.w3schools.com/react/react_usememo.asp

Áp dụng:

- sử dụng useMemo => tránh re-render

- viết theo phong cách typescript

ref: <https://stackoverflow.com/a/43726702>

React.RefObject<HTMLInputElement>

- import WaveSurferOptions

#114. Handle Event với useCallback

Tài liệu: https://www.w3schools.com/react/react_usecallback.asp

- giới thiệu về useCallback

#115. Add Bar Width

Tài liệu: <https://wavesurfer-js.org/examples/?all-options.js>

#116. HTML Canvas Gradients

https://www.w3schools.com/graphics/canvas_gradients.asp

#117. Add Gradients

<https://wavesurfer-js.org/examples/#gradient.js>

<https://wavesurfer-js.org/examples/#soundcloud.js>

gồm: waveColor: (full audio)

progressColor: Hiển thị phần màu của audio đã chạy qua

Lưu ý về nơi định nghĩa gradient => sử dụng document => ko định nghĩa ở level ngoài
định nghĩa type cho function useMemo

Source code video này:

<https://drive.google.com/file/d/1Z9puF1UVBqzMValny34A4EExiYEP8Qs/view?usp=sharing>

#118. Add Time

Tài liệu: <https://wavesurfer-js.org/examples/?soundcloud.js>

sử dụng scss:

npm i --save-exact sass@1.67.0

Source code video này:

https://drive.google.com/file/d/11hBPqQziZB_Uv41FDjFzLKMrvD1iYnHu/view?usp=sharing

#119. Add Hover

Source code video này:

https://drive.google.com/file/d/13_j4PCADN-iaZTL8njmrJc2pMb7XpuTX/view?usp=sharing

#120. Remove ID by Ref

Source code video này:

<https://drive.google.com/file/d/17hFP0KwVm-9sHP0w6aB7DEthYa8WvkGD/view?usp=sharing>

#121. Add Customize Bar

Part 1: Customize Bar + Overlay

Tài liệu:

<https://wavesurfer-js.org/examples/?custom-render.js>

<https://github.com/katspaugh/wavesurfer.js/issues>

Source code video này:

<https://drive.google.com/file/d/1Zc4N77Qpcg-5pQiXsaX37oE2t5FXPoxyg/view?usp=sharing>

//Part 2: todo

Part 3: Design Background + Layout

Source code của video này:

https://drive.google.com/file/d/140UCnFjmg9s_EzRPTv2lvT-eXXyfzkL5/view?usp=sharing

#122. Add Comment

Tài liệu: https://www.w3schools.com/css/css_positioning.asp

Source code video này:

<https://drive.google.com/file/d/1MILC5zKWQkevNCEt9JJo6CcqdCo3YLH9/view?usp=sharing>

Link image: <http://localhost:8000/images/chill1.png>

```
const arrComments = [
  {
    id: 1,
    avatar: "http://localhost:8000/images/chill1.png",
    moment: 10,
    user: "username 1",
    content: "just a comment1"
  },
  {
    id: 2,
    avatar: "http://localhost:8000/images/chill1.png",
    moment: 30,
    user: "username 2",
    content: "just a comment3"
  },
  {
    id: 3,
    avatar: "http://localhost:8000/images/chill1.png",
    moment: 50,
    user: "username 3",
    content: "just a comment3"
  }
]
```

#123. Add Tooltip

Tài liệu:

<https://mui.com/material-ui/react-tooltip/>

Source code video này:

<https://drive.google.com/file/d/15rFUFjl5cn72B5uGcBry5Tk70B3dnE2J/view?usp=sharing>

Chapter 14: Module Auth

Xác thực người dùng với backend và tài khoản mạng xã hội (social media) sử dụng hệ thống với Next.JS và Next-auth.

#124. Tài liệu về Authentication với Next.js

Có 2 trang tài liệu:

Now: (Next auth)

<https://next-auth.js.org/>

<https://www.npmjs.com/package/next-auth>

=> dành riêng cho Next.js

Future: (AuthJS)

<https://authjs.dev/>

<https://www.npmjs.com/package/@auth/core>

=> tích hợp Nextjs và "các framework khác"

1. Chuyện Công nghệ thay đổi theo thời gian

- Công nghệ thay đổi theo thời gian, nên chạy theo công nghệ, là phí time vô ích.

- Thay vì vậy, chọn 1 version, và học nó. Cái bạn học được, chính là mindset (cách tư duy) để giải quyết vấn đề.

Sau này, khi gặp version mới, bạn học sẽ nhanh hơn rất nhiều.

- Về coding, khi bạn cài đặt "chính xác" version của thư viện, thì nó "luôn chạy đúng", yên tâm nhé.

2. Dùng Next-auth hay Authjs

- **Dùng Next-auth vì:**

+ đã sẵn sàng cho production, có tài liệu rõ ràng

+ Authjs chưa sẵn sàng cho production, và tài liệu còn thiếu rất nhiều

+ Tương lai sẽ update khóa học dùng authjs sau

#125. Giới thiệu về Next-auth

Tài liệu:

<https://next-auth.js.org/>

<https://www.npmjs.com/package/next-auth>

1. Cài đặt

npm i --save-exact next-auth@4.23.1

2. Next Auth

<https://next-auth.js.org/getting-started/introduction>

- Là dự án mã nguồn mở, cung cấp giải pháp authentication cho Next.js app

- Hỗ trợ cơ chế stateful (session) và stateless (JWT - json web token)

- An toàn (Security by default)

#126. Test Next Auth

Tài liệu:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-auth-appdir>

Yêu cầu: đã có tài khoản github (và cài git)

1. Tạo github Client

<https://authjs.dev/getting-started/oauth-tutorial#2-configuring-oauth-provider>

Bước 1:

Go to: <https://github.com/settings/developers>

Bước 2: fill thông tin

Homepage URL= <http://localhost:3000>

Authorization callback URL = <http://localhost:3000/api/auth/callback/github>

Bước 3: Tạo client ID/client secret

2. Tạo file .env

- update file .env

#127. Debugs với Next.js và VSCode

Tài liệu:

<https://nextjs.org/docs/pages/building-your-application/configuring/debugging>

<https://stackoverflow.com/a/73553212>

Lưu ý: source code next-auth-appdir-example đã cấu hình sẵn debug
=> copy sang source code next-final

Bước 1: Tạo file launch.json

Chọn tab Debug -> "create a launch.json file" (mô trường node.js)

hoặc manually:

tạo folder: .vscode -> tạo file launch.json

Bước 2: Ghi đè nội dung

//launch.json

```
{
  // Use IntelliSense to learn about possible attributes.
  // Hover to view descriptions of existing attributes.
  // For more information, visit: https://go.microsoft.com/fwlink/?linkid=830387
  "version": "0.2.0",
  "configurations": [
    {
      "name": "Next.js: debug full stack",
      "type": "node-terminal",
      "request": "launch",
      "command": "npm run dev",
      "serverReadyAction": {
        "pattern": "started server on .+, url: (https?://.+)",
        "uriFormat": "%s",
        "action": "debugWithChrome"
      },
      "autoAttachChildProcesses": true,
    }
  ]
}
```

Bước 3: chạy app tại chế độ debug

Lưu ý về giá trị của biến:

Use-client -> client component, cần debugs trên browser

#128. Config Next-Auth Routes

Tài liệu:

<https://next-auth.js.org/getting-started/example#add-api-route>

<https://next-auth.js.org/configuration/initialization#route-handlers-app>

<https://github.com/nextauthjs/next-auth/pull/6777>

(only support app Router since 09/04/2023)

1. Tạo Routes

Bước 1:

<https://next-auth.js.org/configuration/initialization#route-handlers-app>

Tạo thư mục: /app/api/auth/[...nextauth]

Bước 2:

Tạo file route.ts

/app/api/auth/[...nextauth]/route.ts

Nội dung file:

import NextAuth from "next-auth"

```
const handler = NextAuth({
  ... //authOptions (cấu hình providers)
})
export { handler as GET, handler as POST }
```

2. List Providers (nhà cung cấp)

<https://next-auth.js.org/providers/>

Providers là danh sách các services/dịch vụ dùng để login user

Why providers ?

User login tại hệ thống Providers => không cần sử dụng username/password được tạo sẵn tại hệ thống (tăng độ tin tưởng cho user)

Login với username/password của hệ thống đang có ?

Sử dụng "Credentials Providers"

<https://next-auth.js.org/providers/credentials>

#129. Config Github Provider

Tài liệu:

<https://next-auth.js.org/providers/github>

Mặc định, sau khi setup Next-auth, các **endpoint** sau được "tự động thêm vào"

<https://next-auth.js.org/getting-started/rest-api>

GET /api/auth/signin

Displays the built-in/unbranded sign-in page.

=> đăng nhập sử dụng:

<http://localhost:3000/api/auth/signin>

GET /api/auth/signout

Displays the built-in/unbranded sign out page.

Lưu ý: setup route login/logout cho header (đỡ phải paste link sau này)

Setup:

GITHUB_ID=

GITHUB_SECRET=

#url của app

NEXTAUTH_URL=http://localhost:3000

#secret for jwt => add secret to route.ts

NO_SECRET=

#130. Luồng hoạt động của Next-auth ?

Lưu ý 1: Nextjs với authentication, đang đóng vai trò là backend . Không dùng frontend làm authentication (vì dễ bị hack)

Lưu ý 2: Khi sử dụng Provider, chúng ta đang "không quản lý người dùng", vậy làm sao để mỗi lần F5 (refresh), chúng ta biết được ai là người đang đăng nhập ?

Lưu ý 3: Để test hiệu quả, mở tab ẩn danh : ctrl + shift + n

Về Luồng hoạt động:

Bước 1: Nextjs (backend) + Next-auth => tạo thêm các route cần thiết

Bước 2: User nhấn login: <http://localhost:3000/api/auth/signin>

Bước 3: Next auth redirect to Provider

Quá trình user login được thực hiện tại Provider (tính bảo mật cao/tăng độ tin cậy)

Bước 4: User login thành công "tại Provider", Provider sẽ trigger callback URL (đã cấu hình trên provider)

Bước 5: Nextjs nhận được trigger từ provider => lưu thông tin. Vậy lưu ở đâu ?

Nextjs là con lai (hybrid app), khi chạy cả frontend lẫn backend.

Người dùng f5/refresh page, hành động này xảy ra tại frontend.
Frontend sẽ cần gọi lên backend để lấy thông tin người dùng.

Thông thường, thông tin người dùng sẽ được lưu tại "cookies" (do cookies chỉ được đọc từ server), và lưu theo session_id

không nên lưu "full" thông tin user (vì tiềm ẩn rủi ro).

Tuy nhiên, do chúng ta không "control/kiểm soát" thông tin của user (vì user đang được lưu trữ tại Provides),
nên ở cookies, chúng ta "không thể lưu session_id", vì nếu lưu session_id, truyền cái này lên server, server cũng không có thông tin user để trả về.

Vì vậy, giải pháp đề ra là:

Sau khi nhận được thông tin user từ Provider (bước 5), Nextjs sẽ lưu thông tin này vào cookies, dưới dạng jwt (json web token)

jwt này sẽ có secret (password) để giải mã, và chỉ mình server mới giải mã được.

Mỗi lần người dùng f5/refresh, client gửi lên server cookies (chứa jwt).

Server sẽ decode jwt, lưu thông tin này (users) vào session.

Kể từ lúc này, tất cả logic code sẽ được xử lý dựa vào Session (thứ lưu trong memory của server)

#131. Config Session Cho Client

Tài liệu: <https://next-auth.js.org/getting-started/example#configure-shared-session-state>

Sử dụng: SessionProvider

error Error: React Context is unavailable in Server Components

Lỗi này xảy ra, khi sử dụng Client API trong Server Component

Cách fix:

<https://nextjs.org/docs/app/building-your-application/rendering/composition-patterns#using-context-providers>

Giải pháp:

Tất cả Client API cần sử dụng bên trong "client component", bằng cách dùng "use-client"

Không dùng trực tiếp "use-client", mà dùng gián tiếp qua provider

Bước 1:

Tạo thư mục : lib/auth.provider.tsx

```
//auth.provider.tsx
```

```
'use client';
```

```
import { SessionProvider } from 'next-auth/react';
```

```
// Use of the <SessionProvider> is mandatory to allow components that call  
// `useSession()` anywhere in your application to access the `session` object  
// We use client hydration to establish the <SessionProvider />
```

```
export default function AuthProvider({  
    children,  
}: {  
    children?: React.ReactNode;  
) {  
    return <SessionProvider>{children}</SessionProvider>;  
}
```

Bước 2:

sử dụng AuthProvider trong layout.tsx

Bước 3: Sử dụng useSession hook

Tại "client component"

<https://next-auth.js.org/getting-started/example#frontend--add-react-hook>

sử dụng console.log để check data session

Nếu như chúng ta xóa cookies ? (delete session token)

#132. Config Session cho Server

Tài liệu: <https://next-auth.js.org/configuration/nextjs#in-app-router>

Cách làm này sử dụng cho 'server component', hoặc bất cứ file nào chạy trên server (ví dụ như route handler/gọi api)

```
import { getServerSession } from "next-auth/next"

//setup ...next-auth
import { AuthOptions } from "next-auth";
const authOptions: AuthOptions = {
  secret: process.env.NO_SECRET,
  // Configure one or more authentication providers
  providers: [
    GithubProvider({
      clientId: process.env.GITHUB_ID!,
      clientSecret: process.env.GITHUB_SECRET!,
    }),
    // ...add more providers here
  ],
}

// ...
const session = await getServerSession(authOptions);
console.log(">>> check session server: ", session)
```

#133. Customize Session

//Part 1: Ý tưởng

Hiện tại, session "đang có" chỉ lưu thông tin cơ bản, bao gồm "email, image, name".

Với backend có sẵn (cung cấp api cho nextjs), đang dùng cơ chế stateless với JWT (json web token) để xác thực người dùng.

Để nextjs gọi API backend, cần cung cấp "access_token"

=> lưu thêm các thông tin sau vào session:

```
- "_id": "",  
  "username": "",  
  "email": "",  
  "address": ""  
  "isVerify": "",  
  "type": "",  
  "name": "",  
  "role": "",
```

access_token và refresh_token

=> đây là các thông tin mà backend đang trả về nếu người dùng login thành công.

Tuy nhiên, khi login với Provider, chúng ta không có đủ thông tin như trên, vì vậy, giải pháp đề ra là:

1. Khi user sử dụng Provider để login, và login thành công, chúng ta cần gọi backend. Ở đây, chúng ta cần truyền lên "định danh" của người dùng, như là "email"

2. Backend check "định danh" của người dùng. Nếu chưa tồn tại, tạo 1 account, setup type = "PROVIDER"

ví dụ: tài khoản login với github, sẽ có type = "GITHUB"

tài khoản login thông thường (do hệ thống tự tạo), sẽ có type = "SYSTEM"

Sau khi đã xác định được tài khoản nằm trong database của backend

=> trả về thông tin như thông thường

Ở đây, chúng ta đang lưu dữ liệu ở 2 nơi:

- 1 là Provider (lưu thông tin gốc của user)
- 2 là database backend do chúng ta kiểm soát.

Cần lưu tại 2 nơi, vì ví dụ, người dùng sau khi login sẽ tiến hành thao tác với hệ thống => tạo user dễ quản lý (theo id)

// Part 2: API GET token by Social Media

Handle Logout:

<https://next-auth.js.org/getting-started/client#signout>

POST: <http://localhost:8000/api/v1/auth/social-media>

```
body: {  
  type: string  
  username: string  
}
```

1. Next-auth Options

<https://next-auth.js.org/configuration/options#options>

Mặc định, session của Next-auth sẽ sử dụng "jwt"

<https://next-auth.js.org/configuration/options#session>

Sử dụng callback để handle sự kiện "sau khi login" thành công từ Provider

<https://next-auth.js.org/configuration/options#callbacks>

2. JWT callback

```
jwt({ token, user, account, profile, trigger })
```

- sử dụng trigger === signIn

```
{  
  sử dụng account => provider === account.provider  
  email = token.email  
}
```

//lưu ý: đọc mã nguồn

```
token: JWT
```

```
/**
```

* Either the result of the {@link OAuthConfig.profile} or the {@link CredentialsConfig.authorize} callback.

* @note available when `trigger` is `"signIn"` or `"signUp"`.

=> gán thêm thông tin vào token, token này sẽ được giải mã và gán vào session

3. Session callback

//được gọi khi sử dụng session

//Part 3: Complete login with Github

1. Định nghĩa type

<https://next-auth.js.org/getting-started/typescript#main-module>

//Part 4:

2. API

call api inside jwt callbacks

lưu ý: trigger === "signIn" và provider !== "credentials"

POST: <http://localhost:8000/api/v1/auth/social-media>

```
body: {  
  type: string  
  username: string  
}
```

//modify token => modify session

#134. Login with Credentials

Tài liệu:

<https://next-auth.js.org/providers/credentials>

#135. Nested Layout với Route Group

<https://nextjs.org/docs/app/building-your-application/routing/route-groups>

#136. Bài tập design Login Form

Lưu ý: ko cần làm trang register => admin tạo (user login with social account)

Yêu cầu:

- Sử dụng client component (vì cần handle form/tương tác của user)
- Sử dụng MUI
- Code thuần React state, không sử dụng thêm thư viện nào khác
- Chức năng show password
- Sử dụng thuộc tính **sx** của MUI => code css inside jsx
- **Tự code theo cách hiểu (vì đi làm, sẽ là nhận yêu cầu và tự làm =))**

Output:

- Validate form: username/password không được để trống
- Khi nhấn submit, console.log data username/password

chia layout:

<https://mui.com/material-ui/react-grid/>

<https://mui.com/material-ui/react-grid/#basic-grid>

default breakpoints:

<https://mui.com/material-ui/customization/breakpoints/#default-breakpoints>

<https://mui.com/material-ui/react-text-field/>

<https://mui.com/material-ui/react-button/>

<https://mui.com/material-ui/react-divider/>

Sử dụng icons: <https://mui.com/material-ui/material-icons/>

show/hide password: <https://mui.com/material-ui/react-text-field/#input-adornments>

<https://github.com/mui/material-ui/issues/33470>

#137. Chữa Bài Tập Design Login

Source code video này:

<https://drive.google.com/file/d/1nNudDTI3GFpFCZHFSK1ieWT9PvbcJMyC/view?usp=sharing>

#138. Modify login page with Social Media

<https://next-auth.js.org/configuration/pages>

=> customize next-auth route.

```
add pages:  pages: {  
    signIn: "auth/signin"  
}
```

#139. Modify login page with Credential

Fix bugs khi nhấn login => hard reload /redirect

<https://cloudcoders.xyz/blog/nextauth-credentials-provider-with-external-api-and-login-page/>

-setup : redirect = false

<https://next-auth.js.org/getting-started/client#using-the-redirect-false-option>

=> throw Error

#140. Hiển Thị Thông Báo Lỗi Khi Login

Tài liệu:

<https://mui.com/material-ui/react-snackbar/>

<https://mui.com/material-ui/react-alert/>

<https://mui.com/material-ui/react-snackbar/#customization>

Handle press enter: //todo

Chapter 15: Module Tracks

CRUD Track, đồng thời quản lý file nhạc người dùng upload lên sau khi đã đăng nhập

#141. Bài tập hiển thị danh sách tracks (admin)

Danh sách API:

GET <http://localhost:8000/api/v1/tracks?current=1&pageSize=10>

DELETE <http://localhost:8000/api/v1/tracks/:id>

#142. Chưa bài tập hiển thị danh sách tracks (admin)

//todo

#143. Delete a track (admin)

//todo

#144. Ý Tưởng Design Upload Track

Tạo route: track/upload

Lưu ý: slugs và /upload không bị trùng nhau

Design thành 2 tabs

- + Tab upload audio
- + Tab upload images + fill information

<https://mui.com/material-ui/react-tabs/>

Cài đặt thư viện:

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

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

npm i --save-exact react-dropzone@14.2.3 axios@1.5.0

#145. Input Type File

Tài liệu:

<https://developer.mozilla.org/en-US/docs/Web/HTML/Element/input/file>

1. input type = file

- sử dụng input type = file => upload được file
- single/multiple
- allow file extensions => accept attributes

=> Nhược điểm là giao diện xấu

=> cần customize cho chuyên nghiệp

#146. Tích hợp MUI Tabs

<https://mui.com/material-ui/react-tabs/#basic-tabs>

Tham khảo:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/f4893c49d84f0a9f7423df12317edc05ecbb6dfa/src/app/track/upload/page-copy.tsx>

#147. Tích hợp Drag/Drop

Part 1:

<https://react-dropzone.js.org/#section-basic-example>

CSS:

<https://github.com/react-dropzone/react-dropzone/blob/master/examples/theme.css>

//fix typescript: <https://stackoverflow.com/questions/55728316/file-object-does-not-have-a-path-property-typescript>

//remove <Typography/>
=> update: CustomTabPanel

Tham khảo:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/05f3bdc5f9dbe076c763bc112ae0a987db413529/src/app/track/upload/pageUpload1.tsx>

//Part 2:

- Tích hợp button upload

<https://mui.com/material-ui/react-button/#file-upload>

Lưu ý về preventDefault và stopPropagation

//todo:

add on drop event:

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

Tham khảo:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/05f3bdc5f9dbe076c763bc112ae0a987db413529/src/app/track/upload/pageUpload2.tsx>

#148. Bài tập Design Tab Information

Part 1:

progress: <https://mui.com/material-ui/react-progress/#linear-with-label>

Chia layout: <https://mui.com/material-ui/react-grid/>

btn upload: <https://mui.com/material-ui/react-button/#file-upload>

input: <https://mui.com/material-ui/react-text-field/>

//add props: fullWidth + margin

select: <https://mui.com/material-ui/react-text-field/#select>

//Part 2: chẽ bài tập

Tham khảo:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/769dbf73e153521daaa36cf037505641fc25fa45/src/app/track/upload/pageTab2.tsx>

#149. API Upload Single File

- lưu ý sử dụng fetch:
 - + Truyền header
 - + Truyền access token
- lấy access token from session

#150. Upload file with Axios

Tài liệu: <https://axios-http.com/docs/multipart>

<https://stackoverflow.com/a/42879201>

```
headers: { Authorization: `Bearer ${token}` }
```

#151. Axios with Percents

Tài liệu: <https://github.com/axios/axios/issues/629>

Thêm tham số delay cho header. (đơn vị ms)

Ví dụ: 5000 => delay 5s

Source code video này:

<https://drive.google.com/file/d/1V2nX612of1dtQkwafjwWaMFmWY1y1eeE/view?usp=sharing>

#152. Upload track complete

Link download sample files (audio/images...):

https://drive.google.com/drive/folders/1ca-GLosnx0TEIxlcws7-v7EiO_fIB-p7?usp=sharing

//Part 1:

- chia state

//Part 2

- get data form

- POST: <http://localhost:8000/api/v1/tracks>

//Part 3:

Todo

Source code part 3:

https://drive.google.com/file/d/1y4xQa3x_8665QGVCH-E9Oo1Gzr0EbDit/view?usp=sharing

#153. Add Custom Message

1. Disabled tabs

<Tabs

```
  value={value}  
  onChange={handleChange}  
  aria-label="basic tabs example"  
>  
  <Tab label="Tracks" disabled={value !== 0} />  
  <Tab label="Basic information" disabled={value !== 1} />
```

</Tabs>

2. Tích hợp toast

Original:

<https://github.com/ryohey/use-toast-mui>

Trick: press ":"

Bước 1: Copy paste code

Bước 2: import ToastProvider

Bước 3: useToast Hook => lưu ý update file toast (state === true)

Source code tích hợp thư viện:

<https://drive.google.com/file/d/1ClZRF1cLpjJnpQ1pyKJxnqddCOfgAvO/view?usp=sharing>

#154. Profile Page

- Render with slugs (id user trên url)

get list track

POST:

<http://localhost:8000/api/v1/tracks/users?current=1&pageSize=10>

body: truyền id của user (lấy từ url)

#155. Bài tập render List images

Sử dụng card:

<https://mui.com/material-ui/react-card/#ui-controls>

Chia layout: grid

<https://mui.com/material-ui/react-grid/#spacing>

title click => view detail

button click => play audio at footer

Tham khảo:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/8ea3d6ffff829bbbf8c2829a700b4c29bda6d4fc/src/app/profile/%5Bslug%5D/page.tsx>

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/8ea3d6ffff829bbbf8c2829a700b4c29bda6d4fc/src/components/profile.tracks.tsx>

#156. Customize track footer

add layout, customize css

Tham khảo:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-soundcloud-example/-/blob/af36fb14484ef1d98d67e2f4f36750d866617e6a/src/examples/track/custom.layout.track.tsx>

#157. Quản lý React State Global

<https://react.dev/reference/react createContext>

//todo: add images minh họa

Bài toán:

- Sharing state/props giữa các component mà không muốn "lift-up/passing" từ cha xuống con
- Làm sao để chia sẻ state/props giữa các component không có quan hệ với nhau ?

Giải pháp:

- Tạo global state, tương tự session, bọc ngoài "tất cả" component con.

=> bài toán này có tên là "manage state with react"

1. Các thư viện hỗ trợ manage state

- Redux (Redux toolkit):

<https://redux-toolkit.js.org>

<https://www.npmjs.com/package/@reduxjs/toolkit>

- React Query:

<https://tanstack.com/query/v3/>

- Mobx:

<https://mobx.js.org/README.html>

- Recoil:

<https://recoiljs.org/>

2. Tại sao dùng React Context API

- Không cần cài đặt thêm thư viện, vì đây là 1 api đã tích hợp sẵn trong React

- Dữ liệu lưu trữ là đơn giản => context API giải quyết tốt

- Đối với các bài toán phức tạp => sử dụng thư viện để quản lý (management). React context đơn thuần là "sharing" data

#158. Tích hợp React Context

Part 1

Tài liệu: <https://www.js-craft.io/blog/using-react-context-nextjs-13/>

Part 2: Setup state & function

info:(track)

- Sử dụng với typescript:

```
const TrackContext = createContext<IContext | null>(null);
const { currentTrack, setCurrentTrack } = useTrackContext() as IContext;
```

Source code Part 1 + 2:

<https://drive.google.com/file/d/1bSSLZKzbMn6cWBg9j6J1rgXyHf5oJLUI/view?usp=sharing>

#159. Play/Pause Track with Context

Part 1: check play/pause icons

check play/pause icons

```
{  
  (data._id !== currentTrack._id ||  
   data._id === currentTrack._id && currentTrack.isPlaying === false  
  )  
  &&  
  <IconButton aria-label="play/pause"  
    onClick={(e) => {  
      setCurrentTrack({ ...data,.isPlaying: true });  
    }}  
  >  
  <PlayArrowIcon sx={{ height: 38, width: 38 }} />  
  </IconButton>  
}  
  
{data._id === currentTrack._id && currentTrack.isPlaying === true  
 &&  
 <IconButton aria-label="play/pause"  
 onClick={(e) => {  
   setCurrentTrack({ ...data,.isPlaying: false });  
 }}  
 >  
 <PauseIcon sx={{ height: 38, width: 38 }}  
 />  
 </IconButton>  
}
```

//Part 2:

update footer

<https://github.com/lhz516/react-h5-audio-player/issues/119>

```
if (playerRef?.current && currentTrack?.isPlaying === false) {
    // @ts-ignore
    playerRef?.current?.audio?.current?.pause();
}

if (playerRef?.current && currentTrack?.isPlaying === true) {
    // @ts-ignore
    playerRef?.current?.audio?.current?.play();
}

// 

onPause={() => {
    setCurrentTrack({ ...currentTrack,.isPlaying: false })
}}
onPlay={() => {
    setCurrentTrack({ ...currentTrack,.isPlaying: true })
}}
```

#160. Bài Tập Update view detail

//Part 1:

- Trang profile, click vào title => redirect view detail

<Link

```
style={{  
    textDecoration: "none",  
    color: "unset"  
}}  
href={`/track/${data._id}?audio=${data.trackUrl}`}>  
<Typography component="div" variant="h5" >  
    {data.title}  
</Typography>  
</Link>
```

Lưu ý (fetch data tại client)

API Fetch track by ID:

GET <http://localhost:8000/api/v1/tracks/:id>

//Part 2: sync play/pause

//lưu ý: remove tag use-client inside detail track page => fetching data (infinite loop) => hot reload

Source code video này:

<https://drive.google.com/file/d/1CtN8l4wzqCdM176XnjUJZw0a-e6RbiM3/view?usp=sharing>

Chapter 16: Module Likes, Comments

Hiển thị comment của Tracks và chức năng like Tracks

#161. Bài tập CRUD Comments (Admin)

//todo

#162. Bài tập hiển thị comment trên track

- Get list comments:

POST:

<http://localhost:8000/api/v1/tracks/comments?current=1&pageSize=10&trackId=6507bf9cf423204f73c438ca>

```
const res1 = await sendRequest<IBackendRes<IModelPaginate<ITrackComment>>>({  
    url: `http://localhost:8000/api/v1/tracks/comments`,  
    method: "POST",  
    queryParams: {  
        current: 1,  
        pageSize: 10,  
        trackId: params.slug  
    }  
})
```

#163. Bài tập Component Comments

Part 1: Hiển thị hình ảnh mặc định

Download hình ảnh default:

https://drive.google.com/drive/folders/1ca-GLosnx0TElxwcw7-v7EiO_fIB-p7?usp=sharing

- update avatar in header

- update track background image

src={`\${process.env.NEXT_PUBLIC_BACKEND_URL}/images/\${track?.imgUrl}`}

//fetch comment from server

//remove hard coded comments

//add default images in nextjs

- inside public, add user folder

- add new images: default-user.png, default-github.png, default-google.png

Part 2: Design Component Comments

//add margin to layout

//check day from now

import dayjs from 'dayjs';

import relativeTime from 'dayjs/plugin/relativeTime';

dayjs.extend(relativeTime)

npm i --save-exact dayjs@1.11.10

{dayjs(comment.createdAt).fromNow()}

//fix bug empty footer

thêm điều kiện cho footer

#164. Bài tập add a comment

Part 1: Refresh data

sử dụng : router.refresh

<https://stackoverflow.com/questions/76576630/nextjs-13-updating-server-component-data>

<https://stackoverflow.com/questions/53739908/trigger-client-side-reload-in-next-js>

=> sort: "-createdAt"

Part 2: Jump to moment

add comment with moment

<https://wavesurfer.xyz/docs/classes/wavesurfer.default>

passing wavesurfer to children

moment: Math.round(wavesurfer?.getCurrentTime() ?? 0),

//jump track to moment

<https://github.com/katspaugh/wavesurfer.js/issues/1039>

wavesurfer.seekTo(timeToSet / wavesurfer.getDuration())

Lưu ý: fix hàm calcLeft

const hardCodeDuration = wavesurfer?.getDuration() ?? 0;

- fix text in footer:

<https://stackoverflow.com/questions/17779293/css-text-overflow-ellipsis-not-working>

- đang có bug với hot-reloading , not matching ago..

Warning: Text content did not match. Server: "20 minutes ago" Client: "21 minutes ago"

Source code Component Footer và component Comments:

<https://drive.google.com/file/d/1XbqsvFCjxAZET67hBnhbW7mmGtumZXDX/view?usp=sharing>

#165. Bài tập add likes

Part 1: Design

<https://mui.com/material-ui/react-chip/>

Part 2: add api

Lưu ý: Nếu thấy apis trả kết quả chưa đúng => tải lại backend

//fetch likes:

```
const res2 = await sendRequest<IBackendRes<IModelPaginate<ITrackLike>>>({  
    url: `http://localhost:8000/api/v1/likes`,  
    method: "GET",  
    queryParams: {  
        current: 1,  
        pageSize: 100,  
        sort: "-createdAt"  
    },  
    headers: {  
        Authorization: `Bearer ${session?.access_token}`,  
    },  
})
```

color={trackLikes?.some(t => t._id === track?._id) ? "error" : "default"}

//api plus/minus like

POST

http://localhost:8000/api/v1/likes

nextOption: { cache: "no-store" }

Source code video này:

https://drive.google.com/file/d/1URiQ1Kq5GjSMp_vmoTOBC0MddOQ5rRmg/view?usp=sharing

#166. Add count view

//vào trang xem chi tiết, và nhấn nút play lần đầu tiên => count view + 1

POST: <http://localhost:8000/api/v1/tracks/increase-view>

Body: { trackId: ... }

Chapter 17: Module SEO (Search Engine Optimization)

Tối ưu hóa page để tăng kết quả tìm kiếm trên google với Nextjs SEO

#167. Điều cần biết về SEO

1. Lưu ý

- SEO không liên quan gì tới react (mindset) hay kỹ năng code ngôn ngữ lập trình.
- Nội dung chương này, được chia sẻ theo hiểu biết của cá nhân mình.
- Nếu phát hiện sai sót, welcome ? => update sau

2. Khái niệm về SEO

SEO là viết tắt của "Search Engine Optimization" (tối ưu hóa bộ lọc tìm kiếm), hiểu đơn giản là "tối ưu cho kết quả tìm kiếm Google"

Khi bạn gõ 1 từ khóa trên thanh search của google:

- Nếu SEO tốt, xác suất cao kết quả sẽ hiển thị ở top đầu.
- Không SEO, hoặc SEO không tốt, kết quả hiển thị ở trang nào thì không biết :v

3. Hiểu rõ về SEO

- Chỉ quan tâm tới SEO, sau khi bạn đã code "full 1 website và triển khai nó"

Test SEO tại localhost chỉ là basic. Để có thể tối ưu hóa SEO, cần thời gian.

Nguyên tắc: Sai -> Sửa -> Sai -> Sửa... (không có 1 công thức nào tối ưu 100%).

Điều này tương tự như cách bạn chạy quảng cáo Google, Facebook...

- SEO chỉ là tương đối. cho dù SEO tốt đến mấy, cũng là cách thủ công, đều thua chạy ADs.

trả ADs càng nhiều => đứng top càng lâu

- SEO là free ~.~ Hàng FREE thì ???

#168. Công cụ check SEO

check SEO bằng nhiều công cụ, có thể kể tới như:

1. Công cụ online

Chỉ sử dụng công cụ online khi bạn đã triển khai website production.

https://pagespeed.web.dev/?utm_source=psi&utm_medium=redirect

Google PageSpeed Insights: This tool by Google provides a score for both mobile and desktop versions of your website. It also offers suggestions for optimizing your site's performance.

Website: Google PageSpeed Insights

Google Search Console: Google Search Console offers a suite of tools to help you monitor and improve your website's presence in Google search results. It provides valuable insights into how Google views your site and what issues might need attention.

Website: Google Search Console

2. Công cụ local

- Sử dụng công cụ của browser

Test SEO local (lưu ý chạy tại chế độ build) => lighthouse

- Sử dụng các thư viện ?

Với Nextjs < 13, có thể sử dụng next-seo

<https://www.npmjs.com/package/next-seo>

Với nextjs >=13, dùng trực tiếp các công cụ nextjs hỗ trợ

<https://github.com/garmeeh/next-seo#add-seo-to-page>

#169. Metadata

Next.js cung cấp các API về Metadata, giúp chúng ta có thể cải thiện SEO cũng như khả năng chia sẻ website (qua các mạng xã hội)

Tài liệu:

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

<https://nextjs.org/docs/app/api-reference/file-conventions/metadata>

Các kỹ thuật tối ưu hóa SEO cơ bản bao gồm:

Lưu ý: demo với từng tiêu chí, minh họa với Tiki: <https://tiki.vn/nha-sach-tiki/c8322>

1. Thêm tiêu đề, miêu tả cho web page

//todo: demo

ứng với tag <title> và <meta name='description'> tại <head>

2. Thêm miêu tả, image khi share link qua các mạng xã hội như Facebook,

Twitter..

//todo: demo

- ứng với tag

<meta property="og:title"> (og: open graph meta data)

<meta property="twitter:title">

3. Thêm logo cho website

//todo: demo

Bao gồm các file favicon.icon, apple-icon.jpg, icon.jpg

4. Tạo URL "thân thiện" cho việc search (khái niệm slug)

//todo: demo

5. File hỗ trợ cho bot crawl

Bao gồm: robots.txt, sitemap.xml, JSON-LD, manifest.json

6. Responsive Images với Next Image

<https://nextjs.org/docs/app/building-your-application/optimizing/images>

#170. SEO 1: Tối ưu hóa project

- Nguyên tắc: website tối ưu, load càng nhanh (trải nghiệm của user cao) => google đánh giá cao
=> Điều đầu tiên cần làm, là tối ưu source code của bạn

1. Tối ưu hóa Import với Tree Shaking

<https://web.dev/reduce-javascript-payloads-with-tree-shaking/#what-is-tree-shaking>

- Nguyên tắc: "dùng cái gì", "chỉ import vậy"
=> tối ưu hóa source code bundle (không có **deadcode**)

=> giảm size project => web load nhanh hơn

2. Remove deadcode/dependencies

- Xóa các code không dùng (dựa vào eslint)
- Xóa các dependencies không dùng

#171. SEO 2: Đặt tên Title/Description (Static) cho web page

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/optimizing/metadata#static-metadata>

<https://nextjs.org/docs/app/api-reference/functions/generate-metadata>

1. Nguyên tắc

- Sử dụng file **layout.tsx** hoặc **page.tsx**

- Sử dụng **server component**

=> Nếu tuân thủ theo nguyên tắc chỉ sử dụng "use-client" tại "component" => không có bugs

Metadata không hoạt động với "client-component", tức là component với tag "use-client"

2. Cách sử dụng

Tại file page.tsx, hoặc layout.tsx, export metadata

```
import type { Metadata } from 'next'
```

```
export const metadata: Metadata = {
  title: '...',
  description: '...',
}
```

=> Thông thường sẽ viết trong file "page.tsx", vì file layout.tsx định nghĩa cho nhiều file sử dụng

- Nếu Parent định nghĩa "title", và "child" không định nghĩa => lấy title của parent gán cho child

- Nếu child định nghĩa "title", và "parent" cũng định nghĩa => ưu tiên title của "child"

Vấn đề tồn đọng: Với dynamic route, nội dung của page được render "động". Với cách làm ở trên, tất cả các page "động" đang có "title/description..." giống nhau

#172. SEO 3: Đặt tên Title/Description (Dynamic) cho web page

- Ý tưởng: Gọi API để fetch data

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/optimizing/metadata#dynamic-metadata>

Áp dụng: fetch title/description cho các bài track

#173.1 Cách hosting ảnh FREE (Bonus)

Bước 1: Tạo repository với Github/Gitlab

Link demo: <https://github.com/haryphamdev/sharing-host-files>

Bước 2: commit file "ảnh" -> push to repository

Bước 3: Lấy link ảnh

với Github:

- Copy URL của file
- thêm vào cuối url: "?raw=true"
- enter -> lấy link ảnh

hoặc:

Nhấn chuột phải vào file ảnh => copy "image address" :v

Lưu ý: Github và Gitlab đều giới hạn dung lượng lưu trữ với tài khoản FREE.

Vì vậy, chỉ nên lưu "giới hạn" số lượng file.

Nếu bạn muốn không giới hạn, hãy nghĩ tới các dịch vụ trả phí, ví dụ như Amazon S3

#173.2 SEO 4: Share link website trên Facebook/Twitter

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/optimizing/metadata#dynamic-metadata>

1. Sử dụng openGraph và Twitter object

<https://developers.facebook.com/docs/sharing/webmasters/>

ví dụ:

```
openGraph: {  
  title: 'Hỏi Dân IT',  
  description: 'Beyond Your Coding Skills',  
  type: 'website',  
  authors: ['Hỏi Dân IT', 'Eric'],  
  images: [`https://raw.githubusercontent.com/hoidanit/images-hosting/master/eric.png`],  
},
```

2. Về cách test

- F12 check header của page
- Test thực tế, cần deploy với đường link url (không phải là localhost)
- Có thể test (không deploy) với localtunnel

Tài liệu: <https://theboroer.github.io/localtunnel-www/>

Bước 1: Cài đặt

npm install -g localtunnel

Bước 2: Run project

- Build : **npm run build**
- Chạy local: **npm start**

Bước 3: Expose với localtunnel

lt --port 3000

Bước 4: làm theo hướng dẫn trong đường link url

Bước 5: Share link qua social media

<https://socialsharepreview.com/>

3. More

gợi ý các hướng phát triển:

<https://nextjs.org/docs/app/api-reference/file-conventions/metadata/opengraph-image>

#174. SEO 5: Thêm logo cho website

Tài liệu: <https://nextjs.org/docs/app/api-reference/file-conventions/metadata/app-icons>

Bao gồm các file favicon.icon, apple-icon.jpg, icon.jpg
+ add favicon.ico (32x32)

+ add apple-icon.png (180x180)

#175. SEO 6: Url với slug

Part 1:

- sử dụng slugify

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

Cài đặt thư viện:

npm i --save-exact slugify@1.6.6

hoặc <https://www.npmjs.com/package/slug>

demo: <https://trott.github.io/slug/>

Part 2:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/String/split

#176. SEO 7: Robot.txt, sitemap và manifest.json

Test với: <https://tiki.vn/nha-sach-tiki/c8322>

1. Add robots.ts

Tài liệu: <https://nextjs.org/docs/app/api-reference/file-conventions/metadata/robots>

Tham khảo: <https://tiki.vn/robots.txt>

<https://www.youtube.com/watch?v=AyPAfzcVn2c>

2. Add sitemap.ts

Tài liệu: <https://nextjs.org/docs/app/api-reference/file-conventions/metadata/sitemap>

Tham khảo: <https://www.youtube.com/watch?v=0BMwo7moPBo>

3. Add manifest.json

Tài liệu: <https://nextjs.org/docs/app/api-reference/file-conventions/metadata/manifest>

Tham khảo:

<https://web.dev/add-manifest/>

<https://youtu.be/1VVKhNBeQ?si=viPfWI11q9qfbRnO&t=365>

<https://tiki.vn/manifest.json>

Về progressive web app (PWA):

<https://www.youtube.com/playlist?list=PL4cUxeGkcC9gTxqJBcDmoi5Q2pzDusSL7>

#177. SEO 8: JSON-LD

Tài liệu: <https://nextjs.org/docs/app/building-your-application/optimizing/metadata#json-ld>

Tham khảo: <https://www.youtube.com/watch?v=vioCbTo3C-4>

Linked Data: https://www.youtube.com/watch?v=4x_xzT5eF5Q

Next-script: <https://nextjs.org/docs/pages/building-your-application/optimizing/scripts>

<https://stackoverflow.com/a/73064112>

#178. SEO 9: Next.js Image

Part 1: Local Image

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/optimizing/images>

https://developer.mozilla.org/en-US/docs/Learn/HTML/Multimedia_and_embedding/Responsive_images

Download file sample:

<https://drive.google.com/drive/folders/1Nk78UlnH0kwZJMMX9G8cHiFtDhq7HoMd?usp=sharing>

Responsive:

<https://nextjs.org/docs/app/building-your-application/optimizing/images#responsive>

Lưu ý về Image Optimization: <https://vercel.com/docs/image-optimization/limits-and-pricing>

Tham khảo:

<https://www.youtube.com/watch?v=ZKG8JBdgSos>

Part 2: Remote Image

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/optimizing/images#remote-images>

```
<div style={{
    position: "relative",
    height: "150px",
    width: "100%"
}}
>
<Image
    alt="eric image"
    src={`${process.env.NEXT_PUBLIC_BACKEND_URL}/images/${track.imgUrl}`}
    // width={500}
    // height={500}
    fill
    style={{
        objectFit: 'contain',
    }}
/>
</div>
```

Part 3: Responsive Image với Loaders

Tài liệu:

<https://react-slick.neostack.com/docs/example/responsive>

<https://nextjs.org/docs/app/building-your-application/optimizing/images#loaders>

Ví dụ:

https://res.cloudinary.com/ugwutotheeshoes/image/upload/bo_10px_solid_rgb:f78585_e_blur:290,b_rgb:e1e6e9,c_scale,r_10,h_280,w_450/v1632752254/eatery/item-8.jpg

<https://cloudinary.com/blog/optimize-images-in-a-next-js-app-using-nextimage-and-custom-loaders>

```
responsive: [
  {
    breakpoint: 1024,
    settings: {
      slidesToShow: 3,
      slidesToScroll: 3,
      infinite: true,
      dots: true
    }
  },
  {
    breakpoint: 600,
    settings: {
      slidesToShow: 2,
      slidesToScroll: 2,
      initialSlide: 2
    }
  },
  {
    breakpoint: 480,
    settings: {
      slidesToShow: 1,
      slidesToScroll: 1
    }
  }
]
```

Part 4:

Lưu ý: check score SEO với lighthouse

Chapter 18: Advance - Tối ưu dự án Next.JS

Gọi là advance, vì sau khi code xong muốn tối ưu hóa project

#179. Add Progress Bar

<https://www.npmjs.com/package/next-nprogress-bar>

Cài đặt thư viện:

npm i --save-exact next-nprogress-bar@2.1.2

#180. Handle Error

1. Error Page

Tài liệu: <https://nextjs.org/docs/app/api-reference/file-conventions/error>

Link issue: <https://github.com/vercel/next.js/issues/52993>

2. Notfound Page

<https://nextjs.org/docs/app/api-reference/file-conventions/not-found>

<https://nextjs.org/docs/app/api-reference/functions/not-found>

#181. Middleware

Tài liệu:

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

1. Sử dụng middleware

- Tạo file middleware.ts at root:

Mô hình: request -> middleware -> response

Ví dụ:

```
import { NextResponse } from 'next/server'
import type { NextRequest } from 'next/server'
// This function can be marked `async` if using `await` inside
export function middleware(request: NextRequest) {
  if (request.nextUrl.pathname.startsWith('/home')) {
    return NextResponse.redirect(new URL("/", request.url))
  }
}
// See "Matching Paths" below to learn more
export const config = {
  matcher: ['/home'],
}
```

2. Protected Routes

handle protected page: <https://next-auth.js.org/configuration/nextjs#middleware>

Lưu ý: off middleware default của nextjs

- thêm custom pages trong file route.ts [...nextauth]

```
pages: {
  signIn: "/auth/signin"
}
```

- update file.env: NO_SECRET => NEXTAUTH_SECRET=justARandomString

====

```
export { default } from "next-auth/middleware"
```

```
export const config = { matcher: ["/playlist", "/track/upload"] }
```

#182. Add Loading

Tài liệu:

<https://nextjs.org/docs/app/api-reference/file-conventions/loading>

<https://nextjs.org/docs/app/building-your-application/routing/loading-ui-and-streaming>

Về React Suspense: <https://react.dev/reference/react/Suspense>

<https://react-devtools-tutorial.vercel.app/toggling-suspense-fallbacks>

Bonus về loading Skeleton:

<https://mui.com/material-ui/react-skeleton/>

Lưu ý: Nextjs tự động "router cache", có nghĩa là chỉ load lần đầu, từ lần sau sẽ có data cache

Lưu ý nếu không hoạt động khi f5 : xóa node_modules, và file package_lock.json để test phần loading này

Để test sự delay, add tham số delay vào headers của request, hoặc:

`await new Promise(resolve => setTimeout(resolve, 3000))`

Download file skeleton mẫu:

<https://drive.google.com/file/d/1tLEHa7vDJUki92MBu8FoQGDgLcc4qWue/view?usp=sharing>

#183. Nextjs Build Script

Trước khi chạy build, cần exit project (nếu không sẽ lỗi)

1. Phân biệt môi trường thực thi code

Trong file package.json, chúng ta có:

```
"scripts": {  
  "dev": "next dev",  
  "build": "next build",  
  "start": "next start",  
  "lint": "next lint",  
  "post-update": "echo \"codesandbox preview only, need an update\" && yarn upgrade -  
  latest"  
},
```

- Chạy npm run dev => thực thi: next dev (chạy project tại localhost, chế độ development)

- Chạy npm run build => thực thi: next build => build project

- Chạy npm run start /npm start => chạy kết quả vừa build

- Môi trường dev:

+ code typescript => khi chạy, cần dịch code từ typescript sang javascript
+ để có hot reloading, và tăng tốc độ => cần lưu code dịch (từ ts sang js) vào RAM
=> thông thường, sẽ chậm trong lần đầu vào giao diện

- Môi trường production:

+ đã dịch "tất cả" code từ ts sang javascript => ko cần tới RAM
+ tốc độ nhanh nhất (vì code là final và "đã build")

2. Quá trình build nextjs

- Bước 1: đã code xong dự án

- Bước 2: chạy câu lệnh: **npm run build**

mục đích của câu lệnh này, là dịch "tất cả code từ typescript sang javascript", vì browser chỉ hiểu js

Input: source code trong thư mục src/thư mục public/node_modules

Output: thư mục .next

- Bước 3: test/preview dự án đã build, bằng câu lệnh: npm start

Lưu ý: khi chạy tại chế độ build (bước 3), nếu chúng ta sửa đổi code, sẽ không có "hot reloading"

muốn xem sự thay đổi => cần build lại

muốn hot reloading => chạy tại chế độ dev

#184. Build với Docker

Part 1:

Tài liệu:

<https://docs.google.com/document/d/1pKaG3mwLpmUoymyiiBgGRl2e2uCOOXcoS9vpd44l1o/edit>

update tất cả http://localhost:8000 => thành file .env
\${process.env.NEXT_PUBLIC_BACKEND_URL}

update .env.development
.env.production

Bước 2:

update next.config.js

```
output: "standalone",
hostname: 'host.docker.internal',
port: '8001',
```

Part 2:

Download file mẫu:

<https://drive.google.com/file/d/1tKRy2B0LvRSjex3U426VGdQkqcaS55PI/view?usp=sharing>

Bước 3: download docker folder

=> update MONGO URL

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/deploying#docker-image>

Ví dụ với docker:

Docker file:

<https://github.com/vercel/next.js/blob/canary/examples/with-docker/Dockerfile>

với next.config, update:

<https://nextjs.org/docs/pages/api-reference/next-config-js/output#automatically-copying-traced-files>

output: 'standalone',

(lưu ý: chỉ nên setup như trên khi chạy với docker)

If your project uses Image Optimization with the default loader, you must install sharp as a dependency:

npm i sharp

#185. Phân tích kết quả Next.js build

//đọc kết quả build

- λ (Server) server-side renders at runtime (uses getInitialProps or getServerSideProps)
- (Static) automatically rendered as static HTML (uses no initial props)

demo với build web hoidanit

#186. SSR và Static Site Generation

<https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation>

Ví dụ:

SSR: view detail track

SSG: view course web hoidanit.com.vn

Nhắc lại:

CSR: client side rendering: ví dụ waveform

SSR: server side rendering: ví dụ: view track detail

Với SSR: mô hình request -> response

Nếu: webpage tốn nhiều thời gian để tạo nên, vì Server cần tạo ra file .html (on the fly) rồi gửi lại client ?

1. Static Site Generation

SSG là cách "tạo trước file .html" cho webpage.

Client gửi request -> server "gửi lại ngay lập tức" file .html đã được chuẩn bị trước, mà không tốn công "generate"

=> đây gọi là "static" (bị động/tĩnh)

2. Quá trình tạo ra SSG

- Mặc định, với page thông thường, next.js auto tạo SSR (tức là render file .html trên server)

- Đối với dynamic route:

SSR xảy ra trong quá trình build dự án:

Bước 1: "fetch trước" data từ backend (how ?)

Bước 2: generate ra file .html

Bước 3: khi user vào "dynamic routes", thay vì tốn thời gian để build page, server gửi về ngay lập tức file .html đã được build tại bước 2.

#187. SSG và generateStaticParams

<https://nextjs.org/docs/app/api-reference/functions/generate-static-params>

<https://nextjs.org/docs/pages/building-your-application/rendering>

Tham khảo:

warn Entire page ... deopted into client-side rendering

<https://github.com/vercel/next.js/issues/48442>

<https://github.com/vercel/next.js/issues/48442#issuecomment-1510146754>

Test = cách:

Bước 1:

Tạo mới route /test/[slug] => lưu ý: không nằm trong route group

Bước 2: Tạo component với "hardcode slug"

Bước 3: npm run build

=> check .next/server/app ... => check xem có generate trước file html hay không ?

Bước 4: Test với "slug" được tạo sẵn, và slug không được tạo sẵn

Bước 5: move track to outside (with layout)

IMPORTANT => remove // nextOption: { cache: "no-store" }

#188. NextJS Caching

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/caching>

By default (Mặc định), khi bạn build 1 website với Next.js, bạn đang build 1 static site.

Có nghĩa là 1 website gồm các file HTML tĩnh (đã được build từ phía server)

Từ "tĩnh" ở đây, có thể hiểu là "data hiển thị" không thay đổi theo thời gian
=> cơ chế "caching" của Next.js phục vụ đắc lực cho mục tiêu.

Nextjs sẽ "rất tốt và không cần phải cấu hình" nếu bạn build 1 website mà nội dung của nó ít có sự thay đổi, ví dụ:

- personal blog
- website giới thiệu công ty...

Tuy nhiên, sẽ có trường hợp, dữ liệu website thay đổi theo thời gian, ví dụ khi bạn build **e-commerce website**

- giá sản phẩm, số lượng... thay đổi theo thời gian
=> cần can thiệp vào "cơ chế caching" của nextjs

1. Phân loại Cache

<https://nextjs.org/docs/app/building-your-application/caching#overview>

Gồm:

- Request Memoization
- Data Cache
- Full Route Cache
- Router Cache

Hiểu 1 cách đơn giản về cache:

Cache tại client (Browser):

- Mỗi lần điều hướng trang (navigate), chỉ tốn thời gian chờ đợi render lần đầu, từ lần 2 => cached (Router Cache)

Tại server:

- Cùng 1 request, trong cùng 1 page (url), được gọi nhiều lần => auto de-duplicate, tức là chỉ 1 request được thực hiện (Request Memoization)
- với static page, data cho .html được lưu trữ tại các file .json bên trong server (data source) => không cần hit data để lấy data (Data Cache)
- Tất cả các page đều được pre-render trong quá trình build (Full Route Cache)

2. Vấn đề tồn đọng

Vấn đề tồn đọng:

- Nextjs mặc định "caching data", vậy khi data có sự thay đổi cần làm thế nào ? Đặc biệt với SSG (đã build file html, làm sao để update)
- Mỗi lần build project => data sẽ được cập nhật mới nhất ứng với thời điểm đã build ? Sau khi đã build, làm sao để update giao diện, mà ko muốn build lại project :v

#189.1 Revalidating data (Time based revalidation)

<https://nextjs.org/docs/app/building-your-application/data-fetching/fetching-caching-and-revalidating#revalidating-data>

1. Time-based-revalidation

<https://nextjs.org/docs/app/building-your-application/caching>

Ví dụ:

Tạo api: http://localhost:3000/api/test

//route.ts

```
import { NextResponse, NextRequest } from 'next/server'
function randomInteger(min: number, max: number) {
    return Math.floor(Math.random() * (max - min + 1)) + min;
}

export async function GET(request: NextRequest, response: NextResponse) {
    const random = randomInteger(1, 1000000)
    return NextResponse.json({ data: random })
}
```

trong component, gọi api trên

```
const TestA = async () => {
    const res = await sendRequest<any>({
        url: `http://localhost:3000/api/test`,
        method: "GET",
        nextOption: {
            cache: "no-store"
        }
    })
    return (
        <Container sx={{ mt: 5 }}>
            <div>
                {JSON.stringify(res)}
            </div>
        </Container>
    )
}
```

//Các tiêu chí để test:

cache: "no-store" => mỗi lần f5 -> fetch data mới

next: { revalidate: 10 }

=> chờ hết thời gian trên (có thể tắt máy :v)

=> khi gửi request mới (nextjs trigger re-render) => request tiếp theo nhận kết quả mới

#189.2 Revalidating data (On-demand revalidation)

2. On-demand Revalidation

Tài liệu: <https://nextjs.org/docs/app/building-your-application/caching#on-demand-revalidation>

Bao gồm:

- Revalidate Path

<https://nextjs.org/docs/app/building-your-application/caching#revalidatepath>

revalidatePath vs. router.refresh:

router.refresh không clear "Data cache" (chỉ clear router cache và re-render page) => dùng cache: "no-store" :v

- Revalidate Tag

<https://nextjs.org/docs/app/building-your-application/caching#fetch-optionsnexttags-and-revalidatetag>

Part 3:

- Xóa file /test

Áp dụng:

- Fix bugs khi upload new track
- Fix bugs khi like a track

// Test production bằng cách:

Bước 1: Build dự án ở chế độ build

Bước 2: update name track trong database

<https://stackoverflow.com/a/43525665>

{_id: ObjectId('6507bf9cf423204f73c438cc')}

Bước 3: dùng postman gọi api revalidate

Bước 4: F5 giao diện

Chapter 19: Module Playlists

Hoàn thiện tất cả tính năng trong dự án

#190. Bài tập Playlist Page

Tài liệu:

<https://mui.com/material-ui/react-accordion/>

<https://mui.com/material-ui/react-dialog/>

<https://mui.com/material-ui/react-select/>

<https://mui.com/material-ui/react-switch/>

todo list:

- xóa thư mục test
- Thư mục track -> file loading tại [slug]/loading.tsx

Part 1:

Add Empty Playlist:

//component: new.playlist.tsx

- Create an empty playlist:

```
const res = await sendRequest<IBackendRes<any>>({  
  url: `${process.env.NEXT_PUBLIC_BACKEND_URL}/api/v1/playlists/empty`,  
  method: "POST",  
  body: { title, isPublic },  
  headers: {  
    Authorization: `Bearer ${session?.access_token}`,  
  }  
})
```

Part 2:

Add Track to Playlist

- Get list tracks:

```
const res1 = await sendRequest<IBackendRes<IModelPaginate<ITrackTop>>>({  
  url: `${process.env.NEXT_PUBLIC_BACKEND_URL}/api/v1/tracks`,  
  method: "GET",  
  queryParams: { current: 1, pageSize: 100 },  
  headers: {  
    Authorization: `Bearer ${session?.access_token}`,  
  }  
})
```

)

- Get user's playlists:

```
const res = await sendRequest<IBackendRes<IModelPaginate<IPlaylist>>>({  
  url: `${process.env.NEXT_PUBLIC_BACKEND_URL}/api/v1/playlists/by-user`,  
  method: "POST",  
  queryParams: { current: 1, pageSize: 100 },  
  headers: {  
    Authorization: `Bearer ${session?.access_token}`,  
  },  
  nextOption: {  
    next: { tags: ['playlist-by-user'] }  
  }  
)
```

- Update a playlist:

```
const res = await sendRequest<IBackendRes<any>>({  
  url: `${process.env.NEXT_PUBLIC_BACKEND_URL}/api/v1/playlists`,  
  method: "PATCH",  
  body: {  
    "id": chosenPlaylist._id,  
    "title": chosenPlaylist.title,  
    "isPublic": chosenPlaylist.isPublic,  
    "tracks": tracks  
  },  
  headers: {  
    Authorization: `Bearer ${session?.access_token}`,  
  }  
)
```

Source code chức năng playlist:

https://drive.google.com/file/d/11R5PbLBOLb_a6Cbg7Waw1BYR8PW5mMCI/view?usp=sharing

#191. Bài tập Likes page

Lưu ý: đặt tags cho api get liked page (mỗi lần like/dislike a track => revalidate by tag)

```
const res = await sendRequest<IBackendRes<IModelPaginate<ITrackTop>>>({  
  url: `${process.env.NEXT_PUBLIC_BACKEND_URL}/api/v1/likes`,  
  method: "GET",  
  queryParams: { current: 1, pageSize: 100 },  
  headers: {  
    Authorization: `Bearer ${session?.access_token}`,  
  },  
  nextOption: {  
    next: { tags: ['liked-by-user'] }  
  }  
)
```

Source code chức năng Like page:

https://drive.google.com/file/d/1-LKR7hcdGvaXi9gMfB-G0w_bvq_xAhFk/view?usp=sharing

#192. Bài tập Chức năng search

- Tạo page /search

- Logic fetching data viết tại client => vì cần sử dụng searchParams trên url

```
const res = await sendRequest<IBackendRes<IModelPaginate<ITrackTop>>>({  
  url: `${process.env.NEXT_PUBLIC_BACKEND_URL}/api/v1/tracks/search`,  
  method: "POST",  
  body: {  
    current: 1,  
    pageSize: 10,  
    title: query  
  }  
})
```

- update app.header.tsx

```
<StyledInputBase  
  placeholder="Search..."  
  inputProps={{ 'aria-label': 'search' }}  
  onKeyDown={(e: any) => {  
    if (e.key === "Enter") {  
      if (e?.target?.value)  
        router.push(`/search?q=${e?.target?.value}`)  
    }  
  }}  
/>
```

Source code chức năng Search:

<https://drive.google.com/file/d/18bgIBJbzWNvQV7mx2aJ1pigXLnExYRis/view?usp=sharing>

#193. Login with Google

Tài liệu:

<https://next-auth.js.org/providers/google>

Bước 1: Tạo app (project) với Google console

<https://console.cloud.google.com/>

Lưu ý: sau khi tạo xong => switch sang project vừa tạo

Bước 2: Tạo Credentials

Create OAuth client ID

OAuth consent screen (User External)

app domain (lưu ý: không có dấu / ở cuối url)

Authorized JavaScript origins: `http://localhost:3000`

Authorized redirect URIs: `http://localhost:3000/api/auth/callback/google`

Lựa chọn scope: lấy email và profile

Bước 3:

update:

- .env

auth route

auth.signin.tsx (update onclick event)

Bước 4: có thể test với chế độ debug

#194.1 Fix bug dự án

- Active tabs

<https://nextjs.org/docs/app/building-your-application/routing/linking-and-navigating#checking-active-links>

<https://stackoverflow.com/a/76188804>

- Đặt Metadata (title, description, images... cho tất cả các page)

- Refresh token

follow: <https://github.com/nextauthjs/next-auth/issues/1079>

<https://next-auth.js.org/v3/tutorials/refresh-token-rotation>

Nhược điểm: hot-reloading not working :v => sau khi test xong, tăng thời gian của access_token ở phía backend và update file .env với thời gian sống của token lâu hơn

Source code phần refresh_token:

<https://drive.google.com/file/d/1bkqy3MTWz4PEP2TUR1Vu9nU9xeNHwQ8j/view?usp=sharing>

Các bước để test refresh token:

Bước 1: setup backend -> update file .env

JWT_ACCESS_EXPIRE=30s

Lưu ý: cần chạy lại backend để reload file .env

Bước 2: update nextauth api (phần route)

//todo (add source code)

- Tạo new data = cách, sử dụng Vite => delete fake data

- Có bugs => report để mình fix :v

#194.2. Upgrade Next.js Version (và các thư viện khác)

Lưu ý: cài đặt thêm : **npm i sharp**

Kiểm tra version mới nhất của các thư viện, bằng cách sử dụng:

<https://www.npmjs.com/package/npm-check-updates>

Cài đặt:

npm install -g npm-check-updates

Sử dụng: ncu

Test với câu lệnh build: npm run build

====

next: 13.5.1 , 13.5.2, 13.5.3 => lỗi từ 13.5.4

Lưu ý: sử dụng Git để backup code trong trường hợp có lỗi

Nếu không chắc chắn, nên upgrade từng thư viện một, không nên upgrade hàng loạt

====

Lưu ý với next.js 13.5.x

<https://github.com/vercel/next.js/issues/56407>

<https://github.com/vercel/next.js/issues/55682#issuecomment-1728334570>

=> fix bằng cách: **update next.config.js**

```
experimental: {  
  serverMinification: false,  
},
```

Chapter 20: What's next ?

Các định hướng phát triển sau khi đã nắm vững các kiến thức trọng tâm của Next.js

#195. Why NextJS ?

Tham khảo: <https://deno.com/blog/the-future-and-past-is-server-side-rendering>

- Routing

- Rendering:

+ SSR, SSG, ...

- Server component: (hydrate on client)

+ fetching data

+ use sensitive information

+ keep large dependencies on the server

Client component/ Server component Pattern

<https://nextjs.org/docs/app/building-your-application/rendering/composition-patterns#interleaving-server-and-client-components>

#196. Nhận xét về Level Fresher

Nhìn và clone (doanh cần người như vậy), ko phải là basic CRUD =))

Điều bạn cần luyện tập:

- Khả năng tự đọc tài liệu
- Khả năng xử lý lỗi
- Khả năng xây dựng tính năng (đọc tài liệu và tự xử lý lỗi)

Tất cả các điều trên, bạn sẽ luyện tập khi “tự làm project”

IMPORTANT:

KHÔNG SỬ DỤNG CHAT GPT ĐỂ HỖ TRỢ FIX BUG hoặc CODE TÍNH NĂNG.

Hãy nhớ: trước 2023, không có chat GPT.

Và, hãy như 1 đứa trẻ. Bạn yêu cầu nó học tính nhẩm (phép cộng, trừ, nhân, chia).

Thay vì để cho đứa trẻ thực hành, bạn đưa cho nó cái máy tính (chat GPT), liệu rằng, điều đấy là tốt hay xấu ?

Chat GPT không phải là xấu, tuy nhiên, khi bạn chưa đi làm, hãy luyện tập search google và đọc tài liệu bạn nhé.

Đừng có lạm dụng công cụ, khi kỹ năng bạn chưa đủ để sử dụng nó.

**Cài này giống phim trung quốc, nội công chưa đủ, mà đã đòi luyện “tuyệt kỹ võ công”
=> toàn gánh cái bất lợi cho bản thân :v**

#197. Nhận xét về khóa học này

Đây là khóa học “dài nhất” mà mình từng làm. Why ?

1. Mình cần cover lại kiến thức cơ bản của React, vì có nhiều bạn “không học React” từ mình, dẫn tới tình trạng hỏng kiến thức.
2. Ở đây, việc cover React sử dụng với Typescript là nền tảng vững chắc để học Nextjs (vì khi học Nextjs, bạn cần biết React trước)
3. Next.js có quá nhiều khái niệm mới cần cover (so với các version trước đây)

Ưu điểm:

- Đã cover tất cả các kiến thức “trọng tâm” của Next.js 13, gồm có:
 - + **Server component (fetching data)**
 - + **Routing:** dynamic route, group route
 - + **Các kỹ thuật render của Next.js:** SSR (server side rendering), CSR (client side render), SSG (static site generation), ISR (Incremental Static Regeneration)
 - + Sử dụng **Next-auth** để xác thực người dùng
 - + Kỹ thuật **SEO** cơ bản
- **Tích hợp sử dụng MUI ở mức cơ bản nhất** để làm giao diện

Nhược điểm:

- Chưa chuyên sâu vào sử dụng MUI

#198. What's next - Học gì tiếp theo ?

1. Đối với kiến thức của Next.js

Tìm hiểu các chủ đề:

+ Parallel Routes/Intercepting Routes:

<https://nextjs.org/docs/app/building-your-application/routing/parallel-routes>

<https://nextjs.org/docs/app/building-your-application/routing/intercepting-routes>

+ Internationalization:

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

+ Server Actions:

<https://nextjs.org/docs/app/building-your-application/data-fetching/forms-and-mutations>

2. Đối với MUI

- Cách customize component của MUI: <https://mui.com/material-ui/customization/theming/>
- Cách sử dụng dark/light với MUI: <https://mui.com/material-ui/experimental-api/css-theme-variables/migration/#3-prevent-dark-mode-flickering-in-server-side-applications>

3. Keep learning

code backend ?

Level 1:

<https://hoidanit.com.vn/khoa-hoc/backend-restful-server-voi-nodejs-va-express-sql-mongodb-640b539cfe283eefef939870.html>

Level 2:

<https://hoidanit.com.vn/khoa-hoc/nestjs-voi-typescript-mongodb-sieu-de-64686ec6fb456bbb90663dd6.html>

Tự làm dự án thực hành ?

Chapter 21: Update Next.js 14 với Antd (NEW)

Cập nhật version 14 cho Next.js

#199. Chuyện công nghệ update

Tài liệu: <https://endoflife.date/nextjs>

1. End of life

Với Next.js, các mốc thời gian quan trọng:

- **Next.js v12, update lần cuối là v12.3.4 (ngày 21/11/2022)**

=> sau thời gian này là sự ra đời của Next.js 13

Next.js 13 là breaking change, tức là, có rất nhiều cái bạn học từ version 12 trở về trước, "chưa chắc đã đúng", hoặc "làm khác đi/code khác đi" so với Next.js 13

- **Next.js v13, update lần cuối là v13.5.6 (ngày 26/10/2023)**, what the hell :v

=> Nextjs v14 ra đời và KHÔNG CÓ "breaking change" so với Nextjs 13 :v

2. Chuyện công nghệ Update

Bạn chẳng làm gì được chuyện công nghệ nó cập nhật, vì nếu không cập nhật, nó sẽ bị lỗi thời (có nhiều bugs, kém an toàn...)

Vậy chẳng lẽ chạy theo công nghệ liên tục, học liên tục à ???

Trả lời:

- **Chuyện học liên tục là đúng, nhưng không phải lúc nào cũng chạy theo công nghệ.**

Công nghệ sinh ra với mục đích giúp bạn "giải quyết vấn đề". Nếu công nghệ đã giải quyết tốt vấn đề của bạn, thì không nhất thiết phải chạy theo công nghệ.

=> Đây là lý do tại sao có rất nhiều công ty vẫn dùng công nghệ cũ.

Ví dụ: Nhu cầu của bạn là gọi điện thoại => dùng Iphone 5 và Iphone 15 không khác gì nhau, vì đều giải quyết được vấn đề của bạn.

Tuy nhiên, nếu muốn có "dynamic island" => không phải dùng Iphone 15

Giải pháp đề ra:

- Chọn 1 version cụ thể và học nó.
- Sau khi đã học được 1 version rồi => áp dụng vào giải quyết bài toán bạn gặp phải.
Nếu nó không giải quyết được => upgrade version mới.
- Khi bạn học version mới, nó là sự khác biệt so với người mới bắt đầu vì:
 - + bạn có "base kiến thức cơ bản", đã làm quen và hiểu các khái niệm cơ bản của công nghệ
 - + quá trình đọc tài liệu và áp dụng nó sẽ nhanh hơn
- => đừng sợ công nghệ thay đổi, mà hãy tập trung vào mục đích của bạn.
Học 1 cái và giải quyết vấn đề gấp phải.

#200. Một vài lưu ý về Next.js Update

Câu hỏi 1: Tại sao mình không hướng dẫn Next.js 12 ?

- Do **Next.js 13 là breaking change so với Next.js 12**, khi giới thiệu "App Router" , Next.js 12 là "Page router"

Tại sao lại có sự thay đổi trên:

1. Trending đang diễn ra là "server component", có nghĩa là ưu tiên "render dữ liệu React" tại server => tăng trải nghiệm của người dùng
2. React 18 ra đời hỗ trợ rất tốt cho "server component"
3. React recommend Next.js trên trang tài liệu (bỏ Create-React-App) và có sự hợp tác giữa team React & Vercel

Sự hợp tác này giúp Vercel có thêm nguồn lực "chất lượng" và gần hơn với phong cách của React:

- + Bản chất của code là function (không cần OOP ở đây, tức là class). React Hook là ví dụ
- + Nếu tuân theo nguyên tắc trên, code chỉ là function, thì Next.js cần rewrite lại phong cách viết code

Với Next.js 12, khi định nghĩa 1 page:

```
export default function Page({ data }) {  
  // Render data...  
}  
  
export async function getServerSideProps() {}  
  
export async function getStaticProps() {}  
  
export async function getStaticPaths() {  
}
```

Ngoài function Page để render ra data, chúng ta sẽ viết các function để fetch data từ server... một cách "riêng biệt".

Với Next.js 13, quan niệm, code chỉ là function

=> gộp tất cả vào một và giới thiệu "**async component**" và server component.

```
export default async function Page({ data }) {  
  //fetching data first  
  // Render data...  
}
```

=> all-in-one, đúng như cách chúng ta thường code React. Tất cả code chỉ là function.

Như vậy, với Next.js, không chỉ là cập nhật kiến thức mới (app router/server component),

mà là phong cách code/cách tư duy đã khác so với Next.js

=> đã học Next.js thì học từ version 13 bạn nhé.

Câu hỏi 2: Cách hạn chế lỗi khi cài đặt Next.js ?

Chuyện công nghệ update chỉ là sớm hay muộn, và chúng ta không thể biết được khi nào thư viện update

=> trong quá trình hướng dẫn, mình luôn cài đặt "chính xác 1 version cụ thể" => để hạn chế lỗi tối đa.

Ví dụ:

nên làm: npm i --save-exact next@13.5.3

không nên làm: npm i next => với câu lệnh này là cài đặt version nextjs "mới nhất"

Nếu bạn chạy câu lệnh trên, vào ngày 26/10/2023 => bạn cài next.js v13.5.5

cài vào ngày 27/10/2023 => bạn cài next.js v14.0.0 => có breaking change và gây ra bugs:v

#201. Next.js 14 có gì hot ?

So sánh:

<https://nextjs.org/blog/next-13>

<https://nextjs.org/blog/next-14>

So với Next.js 13, điểm nhấn "duy nhất" của Next.js 14.0.0 là: Server Actions (Stable)
Server Actions được giới thiệu từ Next.js 13, tuy nhiên là version alpha, beta...

1 tính năng được đánh dấu là "stable", có nghĩa rằng nó đã sẵn sàng cho production (sau khi đã testing).

Next.js 14 không giới thiệu breaking change

=> tất cả kiến thức học được từ Next.js 13 sử dụng "giống hệt" so với Next.js 14

Góc nhìn thực tế của bản thân mình, dựa vào github để đánh giá "kết quả":

<https://github.com/vercel/next.js/issues>

Cảm giác team Next.js đang chạy "KPI" để ra version mới, trong khi chất lượng "chưa thật tốt".

Với Nextjs 13 giới thiệu breaking change, bao gồm "App Router và Server Component", đồng thời phong cách code cũng khác đi so với next.js 12.

Điều này dẫn tới số lượng Issue của Next.js report bugs duy trì ở ngưỡng 2k (chỉ thấy tăng, chưa thấy giảm)

=> Số lượng Open Issue phản ánh "chất lượng" của sản phẩm

và, có 1 điều "không hay" của Vercel, đây chính là vercel xóa "issue/discussion" complaint về Next.js 13, 14 @@

Mình chia sẻ như trên để chúng ta có 1 góc nhìn khách quan và đánh giá công bằng. cái gì hay thì nên học, chứ không nhất thiết phải là mới nhất, hot nhất.

=> đây là lý do chúng ta nắm vững các kiến thức cốt lõi của Next.js 13 và không vội vã học "server actions" :v

Kiến thức cốt lõi của Next.js 13 chỉ gói gọn: App router và Server Component :v

#202. Setup Next.js 14 Only

Yêu cầu: Node.js version >= 18.17

Source code: <https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter>

Lưu ý: khi thực hiện khóa học, bạn vui lòng clone code để đảm bảo version cài đặt là giống nhau

Bạn có thể tự tạo dự án để có version mới nhất

=> chỉ thực hiện sau khi đã kết thúc khóa học, vì có bugs bạn sẽ cần tự fix

#203. Setup Antd

Tài liệu: <https://www.npmjs.com/package/antd>

Cài đặt:

npm i --save-exact antd@5.11.1

#204. Bài tập Design Login Form với Antd

Lưu ý: để lấy source code, checkout sang nhánh (branch) **final**

https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter/-/tree/final?ref_type=heads

Source code video này: <https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter/-/commit/4a2dd96868c5931d4b246ae049cc4349c1de3a88>

Tài liệu: <https://ant.design/components/form#examples>

#205. Khái niệm Flick giao diện

<https://github.com/ant-design/ant-design/issues/39891>

Nguyên nhân gây ra lỗi:

Bước 1: Nextjs trả về file HTML pre-render và trả kèm CSS và Javascript riêng lẻ (không nhúng trong file html pre-render)

Bước 2: Client load Antd component (use-client) và load tiếp CSS và Javascript

Ở đây là loading async (bất đồng bộ), không theo thứ tự HTML, CSS và Javascript

=> dẫn tới tình trạng load HTML cơ mà chưa có CSS

Giải pháp đề ra:

- CSS cần được trả ra trên server, và đã nhúng kèm trong file HTML khi pre-render.

Client sẽ nhận lại file HTML (đã có CSS sẵn), nên khi load trang (first load) sẽ không bị tình trạng trên

#206. Setup Antd với Render trên Server

Tài liệu:

<https://ant.design/docs/react/use-with-next#using-app-router>

<https://www.npmjs.com/package/@ant-design/cssinjs>

Cài đặt:

npm i --save-exact @ant-design/cssinjs@1.17.2

So sánh cache của next 13 và next 14.0.0

<https://github.com/ant-design/ant-design/issues/41573#issuecomment-1786493390>

Chapter 22: Server Actions (NEW)

Xử lý data trên form với Nextjs và tính năng Server Action

#207. Tổng quan về Server Actions

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/data-fetching/forms-and-mutations>

Alpha: v13.4

<https://nextjs.org/blog/next-13-4#server-actions-alpha>

Stable: v14.0.0

<https://nextjs.org/blog/next-14#server-actions-stable>

Lợi ích của server actions ?

PHP without reloading page .lol

<https://www.the-art-of-web.com/php/form-handler/>

#208. Handle Form với Server

<https://nextjs.org/docs/app/building-your-application/data-fetching/forms-and-mutations#server-only-forms>

https://www.w3schools.com/html/html_forms.asp

Yêu cầu:

- Sử dụng Server Component
- Tạo form với 2 input: username và password
- Submit và logs data trên server

#209. Loading State và useFormStatus Hook

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/data-fetching/forms-and-mutations#displaying-loading-state>

<https://react.dev/reference/react-dom/hooks/useFormStatus>

Lưu ý:

Không sử dụng useFormStatus bên trong component chứa Form (chỉ sử dụng bên trong child component)

```
<Button  
  type='primary'  
  htmlType="submit"  
  loading={pending}>  
>  
  Submit  
</Button>
```

```
await new Promise(resolve => setTimeout(resolve, 5000));
```

#210. Handling Response Data và useFormState Hook

Tài liệu:

<https://nextjs.org/docs/app/building-your-application/data-fetching/forms-and-mutations#error-handling>

<https://react.dev/reference/react-dom/hooks/useFormState>

```
const res = await fetch(  
  "http://localhost:8000/api/v1/auth/login",  
  {  
    method: "POST",  
    headers: {  
      "Content-Type": "application/json",  
    },  
    body: JSON.stringify({  
      username: formData.get('username'),  
      password: formData.get('password')  
    })  
  })  
  
return await res.json();
```

Source code video này:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter/-/commit/cefee2e53d8ec3f8826a2030320fcad394efb8c2>

#211. Các công cụ sử dụng với Server Actions

Bước 1: Design form html thông thường

Bước 2: Định nghĩa “server actions” cho form, sử dụng ‘use-server’

Bước 3: Xử lý trạng thái của form (loading state) với **useFormStatus** Hook

Bước 4: Xử lý response data với **useFormState** Hook

#212. Call Server Actions from Functions ?

Tài liệu: <https://nextjs.org/docs/app/api-reference/functions/server-actions>

Bài toán:

Có sẵn 1 Client Component, muốn tích hợp Server Actions

Source code video này: <https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter/-/commit/5ce59bb89d8b4be9ab4bbabe2e910902e17c5dee>

Chapter 23: Thực Hành Mutate Data với Server Actions (NEW)

Thực hành để nắm vững việc sử dụng Server Actions thông qua ví dụ thực tế

#213. Design Giao Diện CRUD

Lưu ý: setup thời gian hết hạn access token của backend lên 1000d

Source code giao diện CRUD này:

https://drive.google.com/file/d/1EM6UBLIzWtQ1MCPdRzlvsxcf50bbCVHI/view?usp=drive_link

Việc cần làm:

Test giao diện CRUD users

#214. Bài Tập Create User Actions

Yêu cầu:

- Tạo file actions
- Gọi actions trong client component
- Hiển thị thông báo khi tạo thành công/có lỗi
- Revalidate data

Source code video này:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter/-/commit/ca95b7005e70d717867b3718c4b143b632896589>

#215. Bài Tập Update User Actions

Yêu cầu:

- Gọi actions trong client component
- Hiển thị thông báo khi tạo thành công/có lỗi
- Revalidate data

Source code video này:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter/-/commit/3ff4e5702c733d291bd982b27c765199adfc74dc>

#216. Bài Tập Delete User Actions

Yêu cầu:

- Gọi actions trong client component
- Hiển thị thông báo khi tạo thành công/có lỗi
- Revalidate data

Source code video này:

<https://gitlab.com/public-starter-projects1/01-nextjs-basic-soundcloud/nextjs-14-empty-starter/-/commit/b092f75772316ca78c7b356bdffe99d237026d43>

#217. Update Dự Án SoundCloud Sử Dụng Next.js 14

//update source code dự án soundcloud => sử dụng server actions để revalidate tags :v

Part 1: hướng dẫn upgrade to Nextjs 14:

Cần fix next auth

<https://github.com/vercel/next.js/issues/50870#issuecomment-1598655396>

<https://stackoverflow.com/a/76389403>

Part 2: update chức năng like track

#218. So sánh Server Actions và Route Handler

1. Phân biệt Server Actions và Route Handler

Nhắc lại về Route Handler: api endpoint định nghĩa trên Nextjs

Server Actions:

- Có thể dùng ở Server/Client Component
- Cần định nghĩa actions (với tag **use-server**). Actions là javascript function
- Sử dụng được tất cả tính năng của server : read/delete cookies, revalidate cache, redirect...
- Tính năng đơn giản, Actions có thể định nghĩa trực tiếp trong server component (tập trung code lại 1 nơi)
- Tính bảo mật cao (vì code run trên server)

Route Handler (API định nghĩa trên nextjs)

- Có thể dùng ở Server/Client Component
- Cần định nghĩa endpoint (file **route.ts**), với method POST, GET, PUT, DELETE...
<https://nextjs.org/docs/app/api-reference/file-conventions/route#http-methods>
- Sử dụng được tất cả tính năng của server : read/delete cookies, revalidate cache, redirect...
- Với tính năng đơn giản, "vẫn cần định nghĩa file route.ts" riêng lẻ
- Tính bảo mật cao (vì code run trên server)

Về tính bảo mật: cả route handler và server đều có tính bảo mật cao (vì code run trên server)

2. Ưu điểm/Nhược điểm của server actions

Ưu điểm:

- Tất cả "actions" về bản chất là javascript function (chỉ cần định nghĩa "use server" ở đầu file)

=> tập trung tất cả actions tại một vài file

=> dễ quản lý và sửa đổi

Với API endpoint => càng nhiều endpoint => càng nhiều file route.ts

- Có thể gom nhiều api endpoint vào 1 actions => tiết kiệm được thời gian cho client (vì chỉ gọi đúng 1 endpoint)

- Server Actions có thể chạy mà không cần javascript enabled ở phía client

Nhược điểm:

- không có tính tái sử dụng giữa các dự án khác nhau, vì "actions" gắn liền với code của server.

=> API có tính tái sử dụng (giữa các dự án) là cao hơn.

- Phù hợp nhất khi kết hợp với Next.js làm backend

#219. Tổng kết về Server Actions

1. Giải pháp đề ra

Hiện tại chia có "best practices for Server actions"

(Ví dụ với React, uncontrolled component nên sử dụng thay vì controlled component)

Có thể dùng:

- Client gọi server actions
 - Server actions gọi api endpoints
- => tức là kết hợp giữa server actions và route handler (apis)

=> cách làm này có các lợi thế sau:

- Nextjs không cần làm backend, chỉ đảm nhận vai trò frontend và không quan tâm tới logic của backend
- không cần định nghĩa route.ts (route handler)
- quản lý tập trung actions (tính bảo mật cao, dễ sửa đổi)

Lời Kết

Như vậy là chúng ta đã cùng nhau trải qua hơn 200+ videos về sử dụng framework Next.JS dành cho React (typescript)

Tất cả các kiến thức mình chia sẻ, đều được lấy từ kinh nghiệm đi làm của mình và...

trang tài liệu Next.JS: <https://nextjs.org/docs>

Dĩ nhiên rằng, trong quá trình quá trình thực hiện khóa học này, mình sẽ không thể tránh khỏi những sai sót (vì nếu không có sai sót thì mình làm member của team Nest.JS rồi :v).

Vì vậy, nếu thấy sai sót, các bạn cứ thoải mái đóng góp qua Fanpage Hỏi Dân IT nhé.
<https://www.facebook.com/askITwithERIC>

Nếu bạn thấy khóa học này hữu ích, đừng quên Review đánh giá trên Udemy để nhận được ưu đãi (giảm giá) cho các khóa tiếp theo nhé ^^

Hẹn gặp lại các bạn ở các khóa học tiếp theo
Hỏi Dân IT (Eric)