Chapter 0: Tại Sao Lại Cần Biết Debugs **1**
#0 Download Tài Liệu Khóa Học 1
#1.1 Chuẩn Bị Công Cụ & Môi Trường 1
#1.2 Tại Sao Mình Làm Khóa Học Này 3

Chapter 1: Tư Duy Khi Gặp Bugs & Cách Xử Trí **5**
#2 Google Chrome DevTools 5
#3 Các Level Của Logging 5
#4 Phân Loại Bugs 5
#5 Tư Duy Khi Gặp Bugs 6
#6 Bài Tập Về Bugs (Basic) 8
#7 DevTools - The Sources Panel 8
#8 Sử Dụng Câu Lệnh Console.log 9
#9 Sử Dụng Debugger 9
#10 Sử Dụng Breakpoints 9
#11 Watch Expressions - Xem Giá Trị Của Biến 9
#12 Thực hành sử dụng Watch Expressions 9
#13 Conditional Breakpoints 9
#14 Run debugger in Vscode ? Why not ? 10
#15 Bài tập debugger (Advance with React) 10
#16 Chữa Bài Tập Debug 1 10
#17 Chữa Bài Tập Debug 2 10
#18 Các Bước Thường Làm Khi Gặp Lỗi 11

Chapter 2: Debug Ứng Dụng Backend Node.JS **12**
#19 Tổng Quan Về Công cụ Debug cho Backend 12
#20 Debug Ứng Dụng Backend Như Thế Nào ? 13
#21 Debug với Ứng Dụng Node.JS Đơn Giản(dạng Basic) 13
#22 Debug với Ứng Dụng Node.JS Thực Tế (dạng Advance) 14
#23 Debug Với Ứng Dụng Node.JS Trong Lộ Trình Udemy Fullstack 17

Chapter 3: Quản Lý Node.JS Packages Hiệu Quả **18**
#24 Ý Nghĩa của SemVer - Đặt Lịch Sử Version 18
#25 Câu Lệnh NPM Install 19
#26 Fix Lỗi với Yarn 20

TỔNG KẾT **22**

Chapter 0: Tại Sao Lại Cần Biết Debugs

#0 Download Tài Liệu Khóa Học

Đối với khóa học này và tài liệu này, các bạn download trực tiếp trên Udemy nhé.

Tất cả tài liệu sử dụng trong các video, mình đều upload và ghi trực tiếp link download trong từng video bài học.

#1.1 Chuẩn Bị Công Cụ & Môi Trường

1.Visual Studio Code (VSCode)
Link download VSCode: https://code.visualstudio.com/download

2.Node.JS
Download version mới nhất của Node.JS: https://nodejs.org/en/download/

Download version v14.7.0 (giống mình): https://nodejs.org/download/release/v14.17.0/

3.Google Chrome
Download Google Chrome: https://www.google.com/chrome/

4. Khi Có Câu Hỏi, hoặc gặp lỗi cần support thì làm thế nào ?

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

Fanpage Chính Thức của Hỏi Dân IT: https://www.facebook.com/askITwithERIC

Lưu ý: các bạn học viên mua khóa học trực tiếp trên Udemy (thanh toán qua Paypal/Visa/Master), thì chủ động inbox qua facebook ở trên cho mình, mình sẽ add các bạn vào group riêng dành cho học viên Udemy.
Khi gặp lỗi trong quá trình học tập, có câu hỏi cần giải đáp, thì các bạn đăng lên group Facebook dành cho học viên Udemy, và inbox trực tiếp cho mình qua Fanpage Hỏi Dân IT để được hỗ trợ.

Với các học viên mua "lậu" khóa học của mình (qua các kênh "hack" Udemy), không mua trực tiếp qua mình (Fanpage Hỏi Dân IT) và "Udemy" thì:

- Mình hi vọng các bạn ý thức được hành động của các bạn đang làm. Hãy biết trân trọng thành quả người khác làm ra, đặc biệt là các thành quả phải "vắt óc" suy nghĩ. Tất cả là chất xám, chứ không phải nằm ngủ mà tạo ra được.

- Udemy và Facebook nó đều ghi logs lịch sử, thành ra ai mua, mình đều biết. Khi các bạn mua lậu (bỏ chi phí cho bọn 'bán lậu'), các bạn đã mất kinh phí, tuy nhiên, lại mất đi quyền lợi của học viên (support 1:1 từ mình trong quá trình học)

- Với khóa học này, mình biết nhiều bạn, đặc biệt các bạn sinh viên sẽ không có điều kiện để tham gia. Mình hiểu cảm giác này, vì mình đã từng qua thời sinh viên giống các bạn. Bạn nào khó khăn quá, thì cứ inbox qua Fanpage Facebook, mình sẽ có giải pháp giúp đỡ các bạn, việc gì phải đi "ăn trộm" như vậy.

- Với phần thực hành của dự án này, mình sẽ tìm cách mã hóa, và chỉ những học viên mua "trong sạch" mới dùng được (kiểu như đánh license)

  Với các bạn mới bắt đầu, mình biết các bạn sẽ không làm được gì, chỉ biết vứt không khóa học không dùng được.

  Với các bạn có trình độ, thì mình tin rằng các bạn biết cái gì nên làm. Ví dụ, mình bỏ 2h để mã hóa khóa học, thì các bạn sẽ dành x2, x3 thời gian đấy để crack khóa học. Liệu nó có đáng để dành thời gian đánh đổi, khi nó chỉ bằng "vài cốc cafe" các bạn uống hàng ngày ???

#1.2 Tại Sao Mình Làm Khóa Học Này

Một tình trạng mình hay gặp phải, đấy chính là các bạn (học viên Udemy và Youtube) hay chat, hỏi và nhờ mình fix bugs hộ.

Nói thật, mình thấy cách làm như vậy nó hơi sai sai… Có lẽ sau này, trên youtube, mình sẽ có 1 video nói về chuyện 'nhờ fix bugs hộ', cũng như lên Facebook hỏi bugs…

Quay về với thực tại, chuyện 'nhờ fix bug hộ'. Với Youtube, mình luôn từ chối, vì đơn giản, mình không có time, còn với Udemy, mình sẽ cố gắng đưa ra giải pháp để các bạn fix, trường hợp bất khả kháng thì mình mới check code và fix (2 bên win-win)

Vậy câu hỏi đặt ra, tại sao mình lại làm khóa học này, một khóa học không có trong dự định (plan) của mình.

Lý do:

- Mình muốn các bạn 'tự fix' trước khi hỏi mình. Khóa học sẽ chỉ cho các bạn công cụ và các bước cần làm.

  Quá trình các bạn tự fix sẽ gồm các giai đoạn chính sau:
  B1: Làm quen với công cụ fix bugs
  B2: Sử dụng công cụ fix bugs (làm theo video mình hướng dẫn)
  B3: **THỰC HÀNH công cụ fix bugs, với chính bugs bạn đang gặp phải**

  Có thể lần đầu tiên các bạn fix bugs nó sẽ tốn thời gian. Tuy nhiên, càng làm nhiều, nó càng dễ. Vì sự thật của việc debugs đấy là 'chúng ta xem code chạy như thế nào', chạy từ dòng nào tới dòng nào, và giá trị của nó đang là gì.

  Nhờ quan sát giá trị của biến (qua debugs). Chúng ta biết được lỗi đang sai ở đâu. Sai thì sửa. Lặp đi lặp lại cho tới khi fix được bugs. Easy.

  Nhờ có bước THỰC HÀNH thì đầu của các bạn với CHỊU NGHĨ . Và khi đấy, kiến thức mới là của các bạn. Các bạn học & hiểu mình đang làm gì.

  Có 'tự mình fix bugs' thì mới tự mình làm project được. Chứ chẳng ai theo các bạn được mãi, right ?

- Các bạn tự fix được bugs thì mình cũng tiết kiệm được thời gian. Thời gian đấy, mình sẽ update khóa học và làm thêm khóa mới. Vẫn là câu chuyện, đôi bên win-win

- Sẽ có những lỗi 'khoai' debugs không giải quyết được, hoặc các bạn mù mờ về cách giải quyết. Đây chính là lúc mình support fix bugs và nó đáng 'để tốn thời gian của cả 2 bên'.

Chapter 1: Tư Duy Khi Gặp Bugs & Cách Xử Trí

#2 Google Chrome DevTools

- Lưu ý về ngôn ngữ hiển thị : English

#3 Các Level Của Logging

https://stackoverflow.com/questions/2031163/when-to-use-the-different-log-levels

Browser:
- Tab Error: Warning + Error
- Tab Warnings
- Tab Info

IDE: (ví dụ khi sử create-react-app/eslint)
https://www.npmjs.com/package/eslint

ESLint là công cụ để phát hiện và hiển thị lỗi trước khi code được chạy.
- Info : hiển thị thông tin khi chạy app (nếu có)
- Warning: hiển thị các cảnh báo (có thể fix/hoặc không, dựa vào cấu hình eslint)
- Error: hiển thị các lỗi (thông thường dẫn tới ứng dụng sẽ không chạy được)

#4 Phân Loại Bugs

1. Lỗi cú pháp (Error with syntax)

2. Lỗi logic

#5 Tư Duy Khi Gặp Bugs

1. Confirm the bug: có bugs hay không ?

đôi khi khách hàng reports: 'app broken' - cái gì đấy không hoạt động/không dùng được ???

- xác nhận sự tồn tại của bugs
- miêu tả được 'quy trình' để tạo ra bugs (gồm các thao tác nào)
- xác nhận bugs 'cần fix'. đôi khi bugs là feature :v

2. gather clues: đi tìm vết của bugs

- Có lỗi gì (error message) ở console hay không ?

Dựa vào thông tin này để 'fix bugs', chứ không phải 'report bugs'. report bugs là công việc của 'tester'
hoặc 'users' (người dùng hệ thống).

Nhiều bạn có thói quen thấy lỗi là đi hỏi, mà không chịu nghĩ xem, đọc xem lỗi nó báo gì, và tại sao nó sinh ra lỗi.
Đi làm, coder không thể code mà không có bugs. nếu code không có bugs, chỉ có thể là máy móc, chứ không phải human.
Bugs là chuyện hoàn toàn bình thường.
Tuy nhiên, việc giải quyết bugs cũng là của coders. (tự thân vận động, không ai giúp được bạn, ngoài bạn và anh Google)

---

but how ?

Nếu lỗi hiển thị ở console của browser => lỗi của javascript => có thể liên quan tới React/HTML/CSS

2 loại lỗi chính:
- lỗi cú pháp : có message error hiển thị ở console
- lỗi logic: không có message error, tuy nhiên, chương trình trả ra kết quả 'khác' với thứ chúng ta mong muốn.

3.Find the problem code: Đi tìm code có vấn đề

Có 2 trường chính minh họa ở bước 2.

- Có error message hiển thị ở console:
+ Đọc lỗi (bắt buộc) : translate xem nó có ý nghĩa gì. Cố gắng suy nghĩ trước
+ Error luôn hiển thị nơi gây ra lỗi (tên file)
-> xác định được file gây ra bugs (đôi khi là dòng nào gây ra lỗi)

- Không có error message/hoặc message ghi chung chung, khó xác định thông tin:
trường hợp này khó hơn, thông thường sẽ được giải quyết = cách xem code chạy như thế nào với debugger/breakpoints

4.Discover Root cause: Tìm tận nơi tạo ra bugs (dòng nào, chữ nào)

Xác định nguyên nhân gây ra lỗi: (why that bugs happening)
- The bugs in your code (tìm được file/dòng code gây ra lỗi)
Thông thường, loại bugs này thường do 'chúng ta' tạo ra trong quá trình code, và đa phần là lỗi cú pháp (syntax).
ví dụ gõ sai tên hàm, biến...

-> fix = cách dựa vào error message và sửa tại dòng đấy.
Kiểm tra giá trị của biến, kiểm tra lỗi cú pháp....

- The bugs is coming from external library or API:
Loại bugs này thường xảy ra khi sử dụng các thư viện (tích hợp thư viện vào project) hoặc sử dụng APIs của backend.

Cách fix:
- Đọc và tìm hiểu về cách sử dụng thư viện (library) tích hợp. Cần phải hiểu công cụ mình sử dụng (hiểu basic)
thì mới fix được. Cách dễ nhất là copy/paste ví dụ của thư viện xem có chạy với dự án của mình không, sau đấy
tiến hành customize, từng bước một.

- Copy lỗi và google (+ tên thư viện): 2 nguồn chính là stackoverflow và github của thư viện (phần issues)
(kỹ năng sinh tồn và giải quyết được 99% bugs khoai :v)

- Nếu lỗi tới từ APIs, cần kiểm tra tab 'network' xem APIs đã hoạt động chưa, đang trả ra dữ liệu thế nào, có đúng với yêu cầu của FE không ?

Với bugs không có error message, chương trình chạy được, nhưng trả ra kết quả sai
-> debug từng bước một, sẽ nói chi tiết thông qua các ví dụ thực hành.

Các bước cần thực hiện khi debugs:
- Với lỗi logic, cần xác định lỗi do FE hay BE.
Lỗi do BE, tức là data từ APIs đang trả ra sai => chỉ cần debugs BE

Nếu do FE thì (tương tự cho trường hợp BE)
Điều đầu tiên cần làm khoanh vùng chức năng lỗi. bao gồm:
- khoanh vùng file gây ra lỗi (ơ đây có thể là 1 hoặc nhiều file)
- khoanh vùng hàm (function) gây ra lỗi

Sau khi đã khoanh được vùng gây ra lỗi thì tiến hành kiểm tra data.
- Với React:
+ kiểm tra data lưu trong state đã đúng chưa
+ kiểm tra data khi render vào JSX đã đúng chưa

=> có thể dùng console.log, debugger hoặc breakpoints để thực hiện các điều trên

#6 Bài Tập Về Bugs (Basic)
Todo...

#7 DevTools - The Sources Panel

Tài liệu: https://developer.chrome.com/docs/devtools/javascript/

Dev tools:
Right click + inspect
Command+Option+I (Mac)
or Control+Shift+I (Windows, Linux)

Gồm 3 thành phần:
+ File Navigator (1) : điều hướng file - tất cả file của project sẽ được list tại đây
+ Code Editor (2): Sau khi chọn file từ (1), source code của file sẽ hiển thị.

+ Javascript Debugging pane (3) : khung hiển thị công cụ debug

(lưu ý về screen size, có thể thứ tự 3 phần trên nó khác)

#8 Sử Dụng Câu Lệnh Console.log
console.log

#9 Sử Dụng Debugger
debugger

#10 Sử Dụng Breakpoints

lưu ý về nơi hiển thị devtool (tùy chọn nằm tại góc bên phải)

+ cách đặt breakpoints.
+ add/remove breakpoints
+ Nút resume => exec breakpoints
+ hover để xem giá trị variables

#11 Watch Expressions - Xem Giá Trị Của Biến
Todo…

#12 Thực hành sử dụng Watch Expressions
Todo…
#13 Conditional Breakpoints
+ tab breakpoints: add/remove

https://developer.chrome.com/docs/devtools/javascript/breakpoints/

https://developer.chrome.com/docs/devtools/javascript/

#14 Run debugger in Vscode ? Why not ?

Ở đây, với FE (frontend), mình dùng trực tiếp Browser để debug.

Còn với IDE, trong chương sau, khi học với BE (backend) mình sẽ hướng dẫn các bạn nhé.

#15 Bài tập debugger (Advance with React)

Bước 1: Clone project
git clone …
Bước 2: Cài đặt thư viện cần thiết
npm i
Bước 3: Chạy project
npm start
Bước 4: Fix Bugs…

fix only syntax:
https://github.com/kristingreenslit/react-debug-exercise

fix syntax + logic:
https://github.com/learn-co-curriculum/react-hooks-practice-debugging-flatipotle

#16 Chữa Bài Tập Debug 1

Todo…

#17 Chữa Bài Tập Debug 2

Todo…

#18 Các Bước Thường Làm Khi Gặp Lỗi

Một vài câu hỏi thường gặp:

- Mình làm đến video X, gọi APIs bị lỗi Y...
- Gọi APIs thấy không phản hồi ngay...

Cách bước cần làm khi có bugs:

Bước 1: cần xác định lỗi do FE hay BE
===
cách sử dụng các tab của devtool: preview/payload...

ngoài response code => check data truyền lên

Bước 2: tiến hành debugs

Bước 3: Lặp lại bước 2 cho tới khi hết bugs

Chapter 2: Debug Ứng Dụng Backend Node.JS

#19 Tổng Quan Về Công cụ Debug cho Backend

Một vài lưu ý khi debug với backend:

1. **Nguyên tắc của debug:**
   Debug là quá trình sử dụng công cụ, xem code chạy từ dòng nào tới dòng nào và đang có giá trị là bao nhiêu. Nếu giá trị của biến 'đang bị sai', thì sửa lại, rồi debug lại. Cứ như vậy cho tới khi nào fix được bugs

2. **Không debug bằng đầu (thông qua suy nghĩ)**
   Thay vì việc nghĩ xem tại sao sinh ra bug (không hành đồng, chỉ suy nghĩ), thì hiện thực hóa suy nghĩ đấy. Tức là, nếu bạn 'hiểu code' chạy từ dòng nào, tới dòng nào, thì đặt breakpoint ngay tại đấy.
   Nhờ có breakpoint, chúng ta sẽ biết được 'giá trị của biến', từ đó sẽ biết được code đang sai ở đâu.

3. **Các công cụ thường được dùng với debug ở backend:**

   Ngoài việc sử dụng cách truyền thống là câu lệnh 'console.log', chúng ta có thể sử dụng công cụ debug có sẵn ở IDE.

   Với VScode, công cụ này hỗ trợ chúng ta công cụ debug tích hợp sẵn.

   Thay vì dùng **'Terminal'** để gõ câu lệnh, chúng ta sử dụng **Tab 'Debug Console',**

   Đồng thời, sử dụng các tiện ích có sẵn như 'Breakpoint', 'Watch'... (tương tự như cách debug với Frontend)

#20 Debug Ứng Dụng Backend Như Thế Nào ?

Trong video, là mình demo với 1 ứng dụng :

- FE chạy React với TypeScript
- BE chạy Java với Spring Boot

Ứng dụng đấy được được tạo bởi JHipster. Một công cụ giúp generate code. Công cụ này nằm ngoài phạm vi của khóa học, nên mình không đề cập.

Mục đích của video này, là cung cấp cho các bạn thấy debug với backend nó sẽ trông như thế nào khi sử dụng thực tế.

Các video tiếp theo sẽ được minh họa với backend chạy bằng Node.js :D

#21 Debug với Ứng Dụng Node.JS Đơn Giản(dạng Basic)

Có 2 dạng project Node.JS khi debug, và nó ảnh hưởng tới cách làm:

- **1 là Project tự viết**
- **Và 2 là Project được dựng theo framework**

Lý do nó khác biệt, vì với 1 framework, thì đôi khi, cộng đồng (community) sẽ phát triển các extension giúp giảm thiểu thời gian debug, tương tự như #20, khi mình debug với Java ngay trong VSCode :D

Còn với project tự viết, thì chúng ta sẽ tự thân vận động thôi, sẽ cần thêm bước đi 'cấu hình' VSCode để nó hiểu cách debug diễn ra như thế nào.

Tài liệu sử dụng:
https://github.com/missating/nodejs-todo

Link backup (trường hợp link github trên bị xóa):
https://github.com/haryphamdev/nodejs-todo-debug-udemy

Tổng quan về debug với Node.JS và VSCode:
https://code.visualstudio.com/docs/nodejs/nodejs-debugging

Video này, chúng ta sử dụng debug với 'Auto Attach'. Hiểu nôm na là không cần cấu hình gì hết, VSCode nó 'tự biết' chạy như thế nào, '**tự động gắn' debug** với quá trình chạy app của chúng ta.

=> Cách làm này chỉ sử dụng cho các dự án cơ bản (dạng basic)

#22 Debug với Ứng Dụng Node.JS Thực Tế (dạng Advance)

1.Cài đặt Database MongoDB

Lưu ý: Chỉ làm bước này, nếu như máy tính của bạn chưa cài đặt MongoDB (hoặc bạn chưa từng dùng mongoDB)

Sử dụng docker compose để cài đặt:

- Yêu cầu: đã cài đặt sẵn ứng dụng Docker Desktop

Link download tài liệu trong video:
https://drive.google.com/drive/folders/1WSUejbXaHtrULE6xw6zfp8uHKbyPFgqj?usp=sharing

Sau khi đã có file docker compose, chạy câu lệnh sau (tại thư mục chưa file docker compose):
docker-compose -p hoidanit-debugger up --build

Sau khi chạy xong, chờ 1 xíu để database Mongo khởi động. Test thử bằng cách:

Truy cập: http://localhost:8081/
username/password: user/123456
=> sử dụng giao diện web để view dữ liệu trong mongodb

2. Tạo dự án backend một cách chuyên nghiệp

Link github repo: https://github.com/hagopj13/node-express-boilerplate

Các bước thực hiện:
B1: npx create-nodejs-express-app <project-name>
Thay project-name = tên dự án bạn muốn

B2: nếu trong dự án (thư mục root) chưa có file .env , thì cần tạo thêm với câu lệnh:
cp .env.example .env

B3: set up env.
Mở file .env và tiến hành update URL của mongo.
Nếu bạn dùng docker thì : (trường hợp tự cài MongoDB không qua docker, các bạn tự điền URL connect nhé)

MONGODB_URL=mongodb://root:123456@localhost:27017/?authSource=admin

Mở file ecosystem.config.json :
update : NODE_ENV : "development"
watch: true

B4: Chạy project với câu lệnh npm start

3. Setup Debug với VSCode

Tài liệu:
https://code.visualstudio.com/docs/editor/debugging

https://code.visualstudio.com/docs/editor/debugging#_launch-configurations

Để debug hiệu quả, chúng ta cần 'tạo file launch.json'

TH1: Debug 'không có logs' ở terminal

Nội dung file launch.json

```
{
    "version": "0.2.0",
   "configurations": [
      {
         "type": "node",
         "request": "launch",
         "name": "Launch Program",
         "skipFiles": [
            "<node_internals>/**"
         ],
         "program": "${workspaceFolder}\\src\\index.js",
         "env": {
            "NODE_ENV": "development"
         },
      }
   ]
}
```

TH2: Debug với logs ở terminal

Nội dung file launch.json

```
{
    "version": "0.2.0",
   "configurations": [
      {
         "name": "Launch via NPM",
         "request": "launch",
         "runtimeArgs": [
            "run",
            "start"
         ],
         "runtimeExecutable": "npm",
         "skipFiles": [
            "<node_internals>/**"
         ],
         "type": "node",
         "envFile": "${workspaceFolder}/.env",
         "env": {
            "NODE_ENV": "development"
         },
      }
   ]
}
```

#23 Debug Với Ứng Dụng Node.JS Trong Lộ Trình Udemy Fullstack

Tài liệu sử dụng:
https://stackoverflow.com/a/50729890

Nội dung file launch.json

```
{
    "version": "0.2.0",
   "configurations": [
      {
         "type": "node",
         "request": "launch",
         "name": "Launch Program",
         "skipFiles": [
            "<node_internals>/**"
         ],
         "program": "${workspaceFolder}/src/server.js",
         "runtimeExecutable": "${workspaceFolder}/node_modules/.bin/babel-node",
         "runtimeArgs": [
            "--nolazy"
         ]
      }
   ]
}
```

Chapter 3: Quản Lý Node.JS Packages Hiệu Quả
#24 Ý Nghĩa của SemVer - Đặt Lịch Sử Version
Tài liệu:
https://semver.org/
https://nodesource.com/blog/semver-a-primer

Semantic version (Ngữ nghĩa/Ý nghĩa - của version), gọi tắt là semver, là quy định về cách đặt tên version dành cho
chương trình phần phần (API/CLI/GUI...)

Cú pháp đánh version:
MAJOR.MINOR.PATCH
ví dụ:
Version 0.3.10 ra đời trước 0.10.3
Version 0.1.1 1.0.0
Version 1.100.100 10.10.10

Thay đổi:
- MAJOR: thường dẫn đến 'breaking change', có nhiều thay đổi đột phá, 'có thể dẫn tới xung đột' với version cũ hơn.
(CRUD với APIs)
- MINOR: thêm các functions tích hợp tương thích với các APIs của version đang có
- PATCH: fix bugs với các APIs hiện tại đang có

ví dụ về version của react. 17.0.2 => 18.0.0
17.0.2 => 17.2.1

17.2.1 => 17.2.6

- lưu ý khi sử dụng thư viện với node.js(file package.json) : range version
"library-x" : "*" -> latest

~ (tilde) and ^ (caret)

"library-x" : "~1.2.3" -> include all patch versions from the one specified up to, but not including, the next minor version
=== 1.2.3 <= x < 1.3.0

"library-x" : "^1.2.3" -> include all patch and minor versions from the ones specified up to, but not including, the next version

=== 1.2.3 <= x <2.0.0

#25 Câu Lệnh NPM Install
https://docs.npmjs.com/cli/v8/commands/npm-install

npm install [<package-spec> ...]
aliases: add, i, in, ins, inst, insta, instal, isnt, isnta, isntal, isntall

eg: npm i moment -> latest

Nếu có file package-lock, or an npm shrinkwrap file, or a yarn lock file
cài đặt theo thông tin trong các file trên.

- thứ tự ưu tiên:
npm-shrinkwrap.json
package-lock.json
yarn.lock

npm install [<@scope>/]<name>:
https://www.c2fo.io/node/npm/javascript/2016/06/03/protecting-your-product-with-npm-save-exact/

#26 Fix Lỗi với Yarn

Lưu ý: Chỉ cần xem và thực hiện video này, nếu sau này cài đặt thư viện bị lỗi
Nếu không có lỗi gì, thì NPM vẫn 'đủ dùng' các bạn nhé :D

Các bước cần làm với npm: node package manager
Cách 1:
Bước 1: clear npm cache với câu lệnh
npm cache clean --force
hoặc
delete tất cả folder tại: %appdata%\npm-cache

Bước 2: xóa thư mục node_modules

Bước 3: cài đặt lại thư viện

Cách 2:
Nếu cách trên không hoạt động thì thử lại:
Bước 0: Cài đặt 'chính xác' version Node.JS như mình sử dụng trong video v14.17.0
Bước 1: thực hiện từ bước 1 tới bước 3 giống như ở trên

Cách 3:
Giải pháp cuối cùng: sử dụng yarn

Lý do yarn hoạt động vì: => tạo 1 cache hoàn toàn mới @@

=> nhanh hơn

Khi sử dụng npm/yarn, các bước 'thực tế' xảy ra là:
Các bước khi cài đặt 1 thư viện mới
ví dụ: npm i react@18.2
B1: kéo thư viện react về máy tính (npm registry)
nó lưu vào đâu ? => node_modules (local)

=> lưu vào thư mục cache

npm i react@18.2 cache => local

Một vài câu lệnh hay sử dụng với yarn:

- Cài đặt yarn sử dụng npm: npm install -g yarn
- Cài đặt thư viện:

Cài tất cả thư viện trong file package.json:

- Với npm, sử dụng npm i hoặc npm install, còn với yarn là: yarn hoặc yarn install

Cài đặt 'chính xác' version của thư viện:
Ví dụ với npm:
npm install --save-exact react-awesome-lightbox@1.8.1

=> dùng với yarn: (các bạn tự convert sang yarn nhé)
yarn add react-awesome-lightbox@1.8.1

Tài liệu:
So sánh yarn và npm:
https://stackoverflow.com/questions/40027819/when-to-use-yarn-over-npm-what-are-the-differences

https://yarnpkg.com/getting-started/qa#is-yarn-faster-than-other-package-managers

Sử dụng yarn:
https://yarnpkg.com/getting-started/usage

TỔNG KẾT

Như vậy là chúng ta đã cùng nhau trải qua một cuộc hành trình về cách debug một ứng dụng từ Frontend tới Backend.

Có thể những cách làm mình chia sẻ với các bạn, nó chưa là tối ưu nhất (trong thực tế), tuy nhiên, nó đã ở mức độ 'đủ dùng' để giúp các bạn có thể vượt qua nỗi sợ khi code nó không chạy.

Với khóa học này, mình mong rằng sẽ giúp bạn có **một bức tranh tổng quát 'về việc cần làm khi code gặp lỗi', thay vì 'tư duy bị động nhờ người khác fix hộ'.**

Dĩ nhiên, thực tế sẽ có những bugs được fix dựa vào kinh nghiệm đi làm, tuy nhiên, đa phần những lỗi các bạn gặp phải, nó nằm ở mặt coding, thì cách debugs trong khóa học này, tuy đơn giản, nhưng lại giải quyết vấn đề một cách triệt để.

Thời sinh viên của mình, chẳng ai 'show' cho mình, cũng như hướng dẫn mình cách debug nó như thế nào. Tất cả các kiến thức mình chia sẻ trong khóa học này, đều được tích lũy trong quá trình mình đi làm. Vì vậy, các bạn yên tâm về 'giá trị' của các kiến thức các bạn học được.

Điều cuối cùng, mình muốn nhắn nhủ tới các bạn, đấy chính là, không ai đi theo các bạn mãi cả.

Sẽ tới lúc các bạn phải 'tự bơi' và đứng trên đôi chân của mình.

Và ở đây**, công cụ debugs chính là một trong những điều giúp các bạn có thể 'sống sót' khi bơi ngoài đại dương kia, tương tự nhưng anh Google** mình hay chia sẻ tới các bạn…

Have fun và hẹn gặp lại các bạn ở các khóa học tiếp theo.

~~~ Tác giả: Hỏi Dân IT ~~~

Về khóa học của Hỏi Dân IT: https://haryphamdev.github.io/hoidanit-udemy/

Liên hệ tác giả: https://www.facebook.com/askITwithERIC/