**Web back-end (1) - Một số khái niệm**

1. Web & Website:

* Web là hệ thống thông tin hoạt động trên Internet.
* Website là tập hợp các trang web có liên quan, được tổ chức dưới một tên miền chung.

1. Ứng dụng Web:

* Là chương trình chạy trên trình duyệt, hoạt động theo mô hình client-server.
* Ví dụ: Facebook, Tiki.

1. Kiến trúc Client-Server:

* Client (máy khách): Trình duyệt web gửi yêu cầu (request).
* Server (máy chủ): Xử lý yêu cầu và gửi phản hồi (response).
* Quá trình duyệt web: Nhập URL → Gửi yêu cầu → Nhận phản hồi → Hiển thị trang web

Quá trình này diễn ra như sau:​

* Người dùng nhập địa chỉ URL vào thanh địa chỉ của trình duyệt.​
* Trình duyệt gửi yêu cầu đến máy chủ web tương ứng.​
* Máy chủ xử lý yêu cầu và gửi lại nội dung trang web.​
* Trình duyệt nhận nội dung và hiển thị trang web cho người dùng.​

***BT Web back-end (1) - Một số khái niệm***

Câu 1.1:

Liệt kê 3 gói thuê bao Internet của 3 nhà cung cấp tại Việt Nam:

1. VNPT – Gói Home 1: 165.000 VNĐ/tháng (tốc độ 30 Mbps)
2. Viettel – Gói Net1 Plus: 185.000 VNĐ/tháng (tốc độ 50 Mbps)
3. FPT – Gói Super 80: 245.000 VNĐ/tháng (tốc độ 80 Mbps)

Câu 1.2:

Xem mã nguồn của trang web <https://react.dev>:

* HTML: Mã đánh dấu cấu trúc trang web (tìm thấy trong file .html hoặc trong thẻ <html>, <body>, <div>,...)
* CSS: Định dạng giao diện trang web (file .css, hoặc nội dung trong thẻ <style>)
* JavaScript: Mã điều khiển tương tác trên trang web (file .js, hoặc trong thẻ <script>)

Ngoài HTML, CSS, và JavaScript, ngôn ngữ phía server có thể được sử dụng để tạo ra trang React.dev.

* Dấu hiệu nhận biết: React.dev là trang tài liệu của React, và React được phát triển bởi Facebook (Meta), thường sử dụng Node.js làm backend.
* Lý do: Khi kiểm tra mã nguồn hoặc gửi yêu cầu đến máy chủ, ta thấy rằng React.dev sử dụng dịch vụ CDN và có thể được triển khai trên Next.js, một framework React chạy trên Node.js ở phía server.

Kết luận: Ngôn ngữ phía server có thể là JavaScript (Node.js).

Câu 1.3:

Phát biểu không đúng về web, trang web và website:

A. Đúng – Web là hệ thống thông tin toàn cầu.  
B. Đúng – Web được tạo thành nhờ các trang web và sự liên kết giữa chúng.  
C. Sai – Một trang web không chứa nhiều website (Website chứa nhiều trang web, chứ không ngược lại).  
D. Đúng – Một website chứa nhiều trang web có liên quan đến nhau.

Đáp án: C

Câu 1.4:

Phát biểu không đúng về website và ứng dụng web:

A. Đúng – Không có ranh giới rõ ràng giữa website và ứng dụng web, vì cả hai đều chạy trên trình duyệt.  
B. Đúng – Cả website và ứng dụng web đều có URL để truy cập.  
C. Đúng – Chúng hoạt động theo mô hình client-server.  
D. Sai – HTML và CSS không đủ để tạo ứng dụng web, vì ứng dụng web cần JavaScript (hoặc ngôn ngữ lập trình backend như PHP, Python, Node.js) để xử lý logic.

Đáp án: D

**Web back-end (2) - Tổng quan về phát triển ứng dụng web**

1. Phát triển ứng dụng web là gì?

Phát triển ứng dụng web là quá trình tạo ra các ứng dụng chạy trên nền tảng web. Quá trình này bao gồm nhiều giai đoạn như:​

* Lập dự án​
* Xác định yêu cầu​
* Phân tích​
* Thiết kế​
* Lập trình​
* Triển khai
* Bảo trì và cập nhật



2. Các phương pháp tạo ứng dụng web

Sử dụng hệ quản trị nội dung (CMS): CMS là phần mềm cho phép tạo, quản lý và chỉnh sửa nội dung trên trang web mà không yêu cầu kiến thức lập trình chuyên sâu. Các CMS phổ biến bao gồm WordPress, Joomla và Drupal. Ưu điểm của CMS là dễ sử dụng, tiết kiệm thời gian và linh hoạt trong việc tùy chỉnh giao diện và chức năng.

Sử dụng framework và thư viện hỗ trợ: Phương pháp này dựa vào các framework và thư viện có sẵn để phát triển ứng dụng web, giúp tăng tốc độ phát triển và đảm bảo tính nhất quán. Ví dụ về các framework phổ biến là Django (Python), Ruby on Rails (Ruby) và Laravel (PHP).​

Lập trình thuần túy (không sử dụng framework): Phương pháp này yêu cầu viết mã từ đầu mà không dựa vào các công cụ hỗ trợ, đòi hỏi kiến thức sâu về ngôn ngữ lập trình và các công nghệ web. Ưu điểm là kiểm soát hoàn toàn mã nguồn và tối ưu hóa hiệu suất, nhưng mất nhiều thời gian và công sức hơn.

***BT Web back-end (2) - Tổng quan về phát triển ứng dụng web***

Câu 2.1:

Trang chủ của các CMS trong lĩnh vực phát triển ứng dụng web:

| Tên CMS | Trang chủ |
| --- | --- |
| WordPress | <https://wordpress.org/> / <https://wordpress.com/> |
| Wix | <https://vi.wix.com/> |
| Joomla | <https://www.joomla.org/> |
| Drupal | <https://www.drupal.org/> |
| Magento | <https://magento.com/> |
| Shopify | <https://www.shopify.com/> |
| Moodle (Hệ thống quản lý học tập) | <https://moodle.org/> |
| Blogger | <https://www.blogger.com/> |

Câu 2.2:

Trang chủ của các framework và thư viện hỗ trợ phát triển ứng dụng web:

| Tên Framework/Thư viện | Trang chủ |
| --- | --- |
| jQuery | <https://jquery.com/> |
| Bootstrap | <https://getbootstrap.com/> |
| D3.js | <https://d3js.org/> |
| React | <https://react.dev/> |
| Angular | <https://angular.io/> |
| Vue.js | <https://vuejs.org/> |

Backend:

| Tên Framework | Trang chủ |
| --- | --- |
| Express.js (Node.js) | <https://expressjs.com/> |
| Django (Python) | <https://www.djangoproject.com/> |
| Flask (Python) | https://flask.palletsprojects.com/ |
| ASP.NET (C#) | <https://dotnet.microsoft.com/en-us/apps/aspnet> |
| Spring Boot (Java) | <https://spring.io/projects/spring-boot> |
| Laravel (PHP) | <https://laravel.com/> |
| Ruby on Rails (Ruby) | <https://rubyonrails.org/> |
| Gin (Go) | <https://gin-gonic.com/> |
| Echo (Go) | https://echo.labstack.com/ |

Câu 2.3:

Có 3 cách để tạo ứng dụng web. Phát biểu nào không đúng?

A. Sử dụng CMS (Ví dụ: WordPress, Shopify, Joomla,...)  
B. Sử dụng framework và thư viện hỗ trợ (React, Angular, Laravel,...)  
C. Viết thuần bằng ngôn ngữ lập trình (HTML, CSS, JavaScript, PHP,...)  
D. Sử dụng trình duyệt web (Sai, trình duyệt web chỉ là công cụ chạy ứng dụng web, không phải cách tạo ứng dụng web)

Đáp án: D

Câu 2.4:

Ưu điểm của CMS. Phát biểu nào không đúng?

A. Cần kiến thức lập trình chuyên sâu (Sai, CMS giúp tạo web mà không cần lập trình chuyên sâu)  
B. Tối ưu SEO (Đúng, CMS như WordPress có hỗ trợ SEO tốt)  
C. Dễ sử dụng, dễ quản lý, tiết kiệm thời gian (Đúng, CMS có giao diện trực quan, dễ dùng)  
D. Chi phí phát triển ứng dụng thấp (Đúng, CMS giúp tiết kiệm chi phí so với phát triển web từ đầu)

Đáp án: A

Câu 2.5:

Ưu điểm của framework và thư viện. Phát biểu nào không đúng?

A. Tăng tốc độ phát triển (Đúng, framework cung cấp sẵn nhiều tính năng)  
B. Cải thiện chất lượng viết mã (Đúng, code theo chuẩn, dễ bảo trì)  
C. Giảm khả năng bảo mật (Sai, framework thường có tính năng bảo mật cao hơn)  
D. Dễ dàng bảo trì và mở rộng (Đúng, framework giúp code có cấu trúc rõ ràng, dễ mở rộng)

Đáp án: C

**Web back-end (3) - Web server**

1. Web server là gì?

Web server có thể được hiểu theo hai nghĩa:​

* Phần mềm: Các chương trình như Apache, Nginx, IIS, Node.js được sử dụng để lưu trữ và cung cấp nội dung cho ứng dụng web.​[Langbiang](https://legiacong.blogspot.com/2021/09/web-back-end-3-web-server.html?utm_source=chatgpt.com)
* Phần cứng: Máy chủ vật lý được cài đặt các phần mềm web server để phục vụ nội dung web.

2. Chức năng chính của web server:

* Lưu trữ tập tin: Lưu trữ tất cả các tập tin và dữ liệu của ứng dụng web, bao gồm HTML, hình ảnh, video.​
* Nhận yêu cầu (request): Tiếp nhận yêu cầu từ trình duyệt web của người dùng khi họ truy cập ứng dụng web.​
* Xử lý yêu cầu: Đọc nội dung tập tin, thực thi mã nguồn, kết nối và làm việc với cơ sở dữ liệu.​
* Gửi phản hồi (response): Trả về kết quả xử lý cho trình duyệt để hiển thị nội dung cho người dùng.

A diagram of a web server

AI-generated content may be incorrect.

1. Người dùng gửi yêu cầu từ trình duyệt

* Khi người dùng nhập địa chỉ URL vào thanh địa chỉ hoặc nhấn vào một liên kết, trình duyệt sẽ gửi một HTTP request đến web server.
* Yêu cầu này có thể là:
  + GET request (lấy dữ liệu, ví dụ: truy cập trang web)
  + POST request (gửi dữ liệu, ví dụ: đăng nhập, gửi form)

2. Web server nhận yêu cầu và xử lý

* Máy chủ web (ví dụ: Apache, Nginx) nhận yêu cầu và kiểm tra loại tài nguyên được yêu cầu:
  + Nếu tài nguyên là một tệp tĩnh (HTML, CSS, hình ảnh, video...), máy chủ sẽ gửi trực tiếp tệp đó về cho trình duyệt.
  + Nếu tài nguyên là một tệp động (PHP, Python, Node.js...), máy chủ sẽ gửi yêu cầu xử lý đến một chương trình hoặc ứng dụng phía server.

3. Gửi yêu cầu đến ứng dụng back-end (nếu cần)

* Nếu yêu cầu cần xử lý dữ liệu, web server sẽ chuyển tiếp yêu cầu đến ứng dụng web (backend), chẳng hạn như một PHP script, Java servlet, hoặc API backend.
* Ứng dụng back-end có thể:
  + Xử lý logic nghiệp vụ.
  + Truy vấn cơ sở dữ liệu.
  + Xử lý xác thực, đăng nhập, thanh toán...

4. Ứng dụng back-end xử lý dữ liệu và gửi phản hồi

* Sau khi xử lý yêu cầu, ứng dụng backend sẽ gửi kết quả về web server.
* Kết quả có thể là:
  + Một trang HTML được tạo động (ví dụ: danh sách sản phẩm, bài viết blog...)
  + Dữ liệu JSON hoặc XML (nếu server hoạt động như API)
  + Mã trạng thái HTTP (ví dụ: lỗi 404 nếu không tìm thấy tài nguyên)

5. Web server gửi phản hồi về trình duyệt

* Web server nhận kết quả từ ứng dụng backend và gửi phản hồi HTTP về cho trình duyệt.
* Phản hồi này có thể chứa:
  + Trang web hoàn chỉnh để hiển thị cho người dùng.
  + Dữ liệu cần xử lý bằng JavaScript nếu là ứng dụng web động.

6. Trình duyệt hiển thị nội dung cho người dùng

* Trình duyệt nhận nội dung HTML, CSS, JavaScript và hiển thị giao diện trang web.
* Nếu trang web có JavaScript, nó có thể tiếp tục gửi các yêu cầu AJAX để lấy thêm dữ liệu mà không cần tải lại trang.

3. Một số phần mềm web server phổ biến:

* Apache​
* Nginx​
* Node.js​
* IIS​
* Tomcat​
* Lighttpd​

4. Cài đặt và sử dụng Nginx:

Bài viết hướng dẫn cách tải, cài đặt và chạy Nginx trên máy tính:​

* Tải Nginx: Truy cập trang chủ [nginx.org](https://nginx.org/), chọn mục Download và tải phiên bản phù hợp với hệ điều hành.​
* Cài đặt Nginx: Giải nén thư mục nginx đã tải về vào ổ đĩa (ví dụ: C:\n​ginx-1.24.0).​
* Chạy Nginx: Nhấp đúp chuột vào tập tin nginx.exe để khởi động chương trình. Kiểm tra trong Task Manager để xác nhận tiến trình đang chạy.​

***BT Web back-end (3) - Web server***

Câu 3.2:

Phát biểu nào không đúng về phần mềm Web Server?

A. Apache – Đúng, Apache là một trong những Web Server phổ biến nhất.  
B. Nginx – Đúng, Nginx là Web Server mạnh mẽ, được dùng rộng rãi.  
C. SSI – Sai, SSI (Server-Side Includes) không phải là Web Server mà chỉ là một tính năng hỗ trợ chèn nội dung động vào trang HTML.  
D. Node.js – Đúng, Node.js có thể được dùng như một Web Server khi kết hợp với Express.js.

Đáp án: C

Câu 3.3:

Phát biểu nào không đúng về chức năng của Web Server?

A. Lưu trữ các tập tin của ứng dụng web – Đúng, Web Server lưu trữ HTML, CSS, JS, hình ảnh, API,...  
B. Nhận request từ client – Đúng, Web Server nhận request từ trình duyệt (HTTP request).  
C. Xử lý request – Đúng, Web Server xử lý request và trả về response.  
D. Nhận response từ máy client – Sai, Client không gửi response, chỉ gửi request, Web Server mới gửi response về.

Đáp án: D

**Web back-end (4) - Ứng dụng web bán hàng**

vòng đời của một dự án gồm các bước:

- Lập dự án

- Xác định yêu cầu

- Phân tích

- Thiết kế

- Lập trình

- Triển khai

- Bảo trì và cập nhật

Môi trường và công cụ sử dụng:

* Ngôn ngữ lập trình phía client:​
  + HTML: Tạo giao diện.
  + CSS: Định dạng và trang trí giao diện.
  + JavaScript: Xử lý hiệu ứng và giao tiếp giữa client và server.
* Ngôn ngữ lập trình phía server:​
  + Ngôn ngữ: JavaScript.
  + Framework: Express.
* Web server:​
  + Node.js.
* Hệ quản trị cơ sở dữ liệu:​
  + PostgreSQL.
* Công cụ làm việc với PostgreSQL:​
  + DBeaver.
* Quản lý mã nguồn:​
  + Git, GitHub.

2. Cài đặt các công cụ cần thiết

* VS Code: Trình soạn thảo mã nguồn được sử dụng để viết và quản lý mã nguồn cho dự án. Hướng dẫn tải và cài đặt tại <https://code.visualstudio.com/>.​
* Git: Hệ thống quản lý phiên bản phân tán, hỗ trợ theo dõi và quản lý các thay đổi trong mã nguồn. Hướng dẫn tải và cài đặt tại <https://git-scm.com/>.

***BT* Web back-end (4) - Ứng dụng web bán hàng**

Câu 4.2:

Lệnh nào được sử dụng để nhúng Git vào thư mục dự án?

A. git init – Đúng, lệnh này khởi tạo một kho lưu trữ Git trong thư mục hiện tại.  
B. git --init – Sai, không có cú pháp này.  
C. git initialize – Sai, Git không có lệnh này.  
D. git embed – Sai, Git không có lệnh này.

Đáp án: A

Câu 4.3:

Lệnh nào sử dụng để kiểm tra trên máy tính đã có phần mềm Git hay chưa?

A. git ver – Sai, không có lệnh này.  
B. git version – Đúng, nhưng không phải cú pháp chuẩn.  
C. git --version hoặc git -v – Đúng, git --version là lệnh chính xác nhất để kiểm tra phiên bản Git.  
D. git --ver – Sai, không có lệnh này.

Đáp án: C

Câu 4.4:

Trong Git, kho lưu trữ (repository) là gì?

A. Là thư mục dự án – Sai, thư mục dự án chưa chắc đã có Git.  
B. Là thư mục dự án đã được nhúng Git – Đúng, khi chạy git init, thư mục trở thành kho lưu trữ Git.  
C. Là thư mục cài đặt phần mềm Git – Sai, thư mục cài đặt Git không liên quan đến repo.  
D. Là thư mục .git (trong thư mục dự án) – Sai, .git là thư mục chứa dữ liệu của repo, nhưng repo không chỉ bao gồm .git.

Đáp án: B

Câu 4.5:

Bạn có thể sử dụng các công cụ sau để viết mã. Phát biểu nào không đúng?

A. Text editor – Đúng, ví dụ như Notepad, Sublime Text.  
B. MS Word – Sai, MS Word không phải là công cụ lập trình. Nó định dạng văn bản không phù hợp với mã nguồn.  
C. IDE, Code Editor – Đúng, ví dụ như VS Code, IntelliJ, Eclipse.  
D. Chatbot (Gemini), Trang web (<https://codepen.io/>) – Đúng, chatbot có thể hỗ trợ viết mã, CodePen giúp thử nghiệm mã trực tuyến.

Đáp án: B

**Git thực hành (1) - Hệ thống quản lý phiên bản**

1. Phiên bản là gì?

Các bản khác nhau của tập tin, thư mục hoặc toàn bộ mã nguồn dự án. Trong quá trình phát triển, mã nguồn liên tục thay đổi, do đó cần có khả năng xem lại hoặc khôi phục trạng thái của dự án tại các thời điểm trước đó.

2. Hệ thống quản lý phiên bản (Version Control System - VCS)

- VCS là phần mềm giúp lưu lại từng thay đổi của mã nguồn dự án và hỗ trợ:​

- Khôi phục trạng thái dự án ở các thời điểm khác nhau.​

- Xác định ai đã thực hiện các thay đổi và nội dung thay đổi.

- Khôi phục nội dung mã nguồn bị xóa

- So sánh những thay đổi của dự án theo thời gian.​

Hệ thống quản lý phiên bản cục bộ: Lưu trữ các phiên bản tại máy tính cá nhân, thường sử dụng công cụ như RCS. Phương pháp này gặp khó khăn khi làm việc nhóm do thiếu khả năng chia sẻ và cộng tác.

Hệ thống quản lý phiên bản tập trung (Centralized Version Control Systems – CVCS): Sử dụng một máy chủ trung tâm chứa các phiên bản của dự án, các máy khách lấy và cập nhật dữ liệu từ máy chủ này.

Hệ thống quản lý phiên bản phân tán (Distributed Version Control Systems - DVCS): Mỗi máy khách lưu trữ một bản sao đầy đủ của kho chứa, bao gồm toàn bộ lịch sử phiên bản. Điều này cho phép khôi phục dữ liệu từ bất kỳ máy khách nào nếu máy chủ gặp sự cố. Ví dụ: Git, Mercurial, Darcs.

***BT* Git thực hành (1) - Hệ thống quản lý phiên bản**

Câu 1.1:

Các đặc điểm của hệ thống quản lý phiên bản cục bộ. Phát biểu nào không đúng?

A. Có thể thực hiện thủ công – Đúng, quản lý phiên bản cục bộ có thể được thực hiện thủ công bằng cách sao lưu tệp.  
B. Không hỗ trợ trong môi trường cộng tác nhiều người – Đúng, vì dữ liệu chỉ nằm trên một máy.  
C. Có thể dùng phần mềm để quản lý phiên bản kiểu cục bộ – Đúng, có thể dùng RCS (Revision Control System).  
D. Các phiên bản của dự án được lưu tập trung trên một máy server – Sai, vì hệ thống cục bộ không có server, tất cả nằm trên máy tính cá nhân.

Đáp án: D

Câu 1.2:

Các đặc điểm của hệ thống quản lý phiên bản tập trung. Phát biểu nào không đúng?

A. Các phiên bản của dự án được lưu tập trung trên máy server – Đúng, vì mô hình tập trung yêu cầu một máy chủ để lưu phiên bản.  
B. Các máy client sẽ chứa tất cả các phiên bản của thư mục dự án cùng với lịch sử thay đổi – Sai, chỉ máy server chứa toàn bộ phiên bản và lịch sử, client chỉ lấy phiên bản mới nhất.  
C. Máy client không thể tải phiên bản của dự án về, khi máy server không hoạt động – Đúng, vì nếu server bị lỗi, client không thể truy cập dữ liệu.  
D. Hỗ trợ làm việc cộng tác nhiều người – Đúng, nhiều người có thể cùng làm việc với dữ liệu từ server.

Đáp án: B

Câu 1.3:

Các đặc điểm của hệ thống quản lý phiên bản phân tán. Phát biểu nào không đúng?

A. Các máy client sẽ chứa toàn bộ các phiên bản của dự án, cùng lịch sử thay đổi – Đúng, hệ thống như Git giúp client có bản sao đầy đủ.  
B. Hỗ trợ làm việc cộng tác nhiều người – Đúng, các client có thể đẩy/pull thay đổi từ remote repository.  
C. Các phiên bản của dự án được lưu trên máy server – Đúng, máy server có thể đóng vai trò làm repository chính.  
D. Bạn không thể tạo và lưu phiên bản khi không có kết nối mạng tới máy server – Sai, hệ thống phân tán như Git cho phép commit cục bộ mà không cần mạng.

Đáp án: D

Câu 1.4:

Tìm trang chủ của các phần mềm quản lý phiên bản:

| Tên phần mềm | Trang chủ |
| --- | --- |
| RCS | https://www.gnu.org/software/rcs/ |
| CVS | https://www.nongnu.org/cvs/ |
| Subversion (SVN) | <https://subversion.apache.org/> |
| Git | <https://git-scm.com/> |
| Mercurial | <https://www.mercurial-scm.org/> |
| Darcs | <http://darcs.net/> |

**Git thực hành (2) - Tổng quan về Git**

1. Git là gì?

Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System - DVCS), cho phép theo dõi và quản lý các thay đổi trong mã nguồn của dự án một cách hiệu quả. Một số tính năng nổi bật của Git bao gồm:​

Theo dõi lịch sử thay đổi

Cộng tác hiệu quả

Phân nhánh và hợp nhất

Bảo mật

Phân tán

2. Tải và cài đặt Git

Để sử dụng Git, cần tải và cài đặt phần mềm này trên máy tính:​

* Truy cập trang chủ <https://git-scm.com/>, chọn phiên bản phù hợp với hệ điều hành đang sử dụng và tải về.​
* Sau khi tải về, thực hiện cài đặt như một phần mềm thông thường.​
* Để kiểm tra Git đã được cài đặt thành công hay chưa:​
  + Mở cửa sổ dòng lệnh (Command Prompt).​
  + Nhập lệnh git --version hoặc git -v và nhấn Enter.​
  + Nếu xuất hiện thông tin về phiên bản của Git, có nghĩa là Git đã được cài đặt thành công trên máy tính.

***BT* Git thực hành (2) - Tổng quan về Git**

Câu 2.2:

Lệnh nào được sử dụng để nhúng Git vào thư mục dự án?

A. git init – Đúng, lệnh này khởi tạo một repository Git trong thư mục hiện tại.  
B. git --init – Sai, không có cú pháp này trong Git.  
C. git initialize – Sai, Git không có lệnh này.  
D. git embed – Sai, Git không có lệnh này.

Đáp án: A

Câu 2.3:

Lệnh nào sử dụng để kiểm tra trên máy tính đã có phần mềm Git hay chưa?

A. git ver – Sai, không có lệnh này trong Git.  
B. git version – Đúng, đây là cách kiểm tra phiên bản Git, nhưng không phải cú pháp chuẩn.  
C. git -v – Đúng, nhưng cú pháp chính xác hơn là git --version.  
D. git --ver – Sai, không có cú pháp này.

Đáp án: C (git --version là chính xác nhất)

Câu 2.4:

Trong Git, kho lưu trữ (repository) là gì?

A. Là thư mục dự án – Sai, thư mục dự án chưa chắc đã là một Git repository.  
B. Là thư mục dự án đã được nhúng Git – Đúng, khi chạy git init, thư mục trở thành repository Git.  
C. Là thư mục cài đặt phần mềm Git – Sai, đây chỉ là nơi Git được cài đặt, không phải repository.  
D. Là thư mục .git (trong thư mục dự án) – Sai, .git chỉ là thư mục ẩn chứa dữ liệu quản lý phiên bản, nhưng repository không chỉ gồm .git.

Đáp án: B

**Web back-end (5) - Nodejs và NPM**

1. Node.js là gì?

Node.js là một môi trường chạy JavaScript mã nguồn mở, đa nền tảng, cho phép phát triển các ứng dụng như máy chủ web, ứng dụng web, công cụ dòng lệnh và các tập lệnh. Node.js không phải là một framework hay ngôn ngữ lập trình, mà là một nền tảng bao gồm nhiều thành phần như:​[Langbiang](https://legiacong.blogspot.com/2021/10/web-back-end-5-lap-trinh-phia-server.html?utm_source=chatgpt.com)

* V8 Engine: Trình biên dịch và thực thi mã JavaScript.​
* HTTP Module: Module cho phép tạo máy chủ HTTP.​
* NPM (Node Package Manager): Công cụ quản lý các gói thư viện và module.

NPM (Node Package Manager)

NPM là công cụ đi kèm với Node.js, dùng để quản lý các thư viện và module trong ứng dụng web. NPM giúp cài đặt, cập nhật và quản lý các gói phụ thuộc một cách hiệu quả.​

Để kiểm tra NPM đã được cài đặt:​

* Kiểm tra phiên bản: Mở cửa sổ dòng lệnh và nhập npm -v. Nếu hiển thị phiên bản của NPM, nghĩa là cài đặt thành công.​

NPM bao gồm ba thành phần chính:​

* Website: Trang web <https://www.npmjs.com/> để tìm kiếm và khám phá các gói thư viện.​
* Công cụ dòng lệnh (CLI): Cho phép cài đặt và quản lý các gói trực tiếp từ dòng lệnh.​
* Registry: Kho lưu trữ các gói thư viện.​

Để sử dụng một gói thư viện, ví dụ như Express, bạn có thể:​[Langbiang](https://legiacong.blogspot.com/2021/10/web-back-end-5-lap-trinh-phia-server.html?utm_source=chatgpt.com)

* Tìm kiếm trên website NPM: Truy cập <https://www.npmjs.com/> và tìm kiếm "express".​
* Cài đặt bằng CLI: Mở cửa sổ dòng lệnh trong thư mục dự án và nhập npm install express để cài đặt Express vào dự án.

***BT* Web back-end (5) - Nodejs và NPM**

Câu 5.2: Nodejs là gì? Phát biểu nào không đúng?

A. Đúng – Node.js là một môi trường (runtime environment) để phát triển ứng dụng.

B. Không đúng – Node.js không phải là ngôn ngữ lập trình, nó sử dụng JavaScript.

C. Đúng – Node.js là một nền tảng phát triển ứng dụng, đặc biệt cho các ứng dụng phía server.

D. Đúng – Node.js có thể tạo web server mà không cần phần mềm như Apache hay Nginx.

Đáp án: B

Câu 5.3: NPM là gì? Phát biểu nào không đúng?

A. Đúng – NPM là viết tắt của *Node Package Manager*.

B. Đúng – NPM giúp quản lý thư viện/phụ thuộc (dependencies) cho ứng dụng Node.js.

C. Đúng – Khi cài Node.js thì NPM cũng được cài theo.

D. Không đúng – NPM gồm website, CLI (Command Line Interface) và registry. Không có GUI chính thức.

Đáp án: D

**Web back-end (6) - Cấu hình Web server trong Nodejs**

1. Tạo tập tin package.json bằng NPM

Mở Terminal hoặc Command Prompt: Sử dụng công cụ dòng lệnh như CMD, PowerShell hoặc Terminal trong VS Code.​[Langbiang+6Langbiang+6Langbiang+6](https://legiacong.blogspot.com/2021/11/web-back-end-6-ket-noi-web-server-va.html?utm_source=chatgpt.com)

 Di chuyển đến thư mục dự án: Sử dụng lệnh cd để chuyển đến thư mục chứa dự án, ví dụ: cd E:\TeoShop.​

 Khởi tạo package.json: Chạy lệnh npm init -y để tạo tập tin package.json với các giá trị mặc định. Nếu thành công, nội dung tập tin sẽ bao gồm các thông tin như tên dự án, phiên bản, điểm khởi đầu (main), các lệnh (scripts), từ khóa, tác giả, giấy phép và mô tả.​

2. Cài đặt và sử dụng Express để tạo Web server

Express là một framework phổ biến cho Node.js, giúp đơn giản hóa việc xây dựng các ứng dụng web. Để cài đặt và sử dụng Express:​

* Cài đặt Express: Trong thư mục dự án, chạy lệnh npm install express để thêm Express vào dự án và cập nhật package.json với thông tin về thư viện này.​
* Tạo tập tin index.js: Tạo một tập tin index.js trong thư mục dự án với nội dung sau:

const express = require('express');

const app = express();

const port = 3000;

app.get('/', (req, res) => {

res.send('Xin chào, đây là trang chủ!');

});

app.listen(port, () => {

console.log(`Ứng dụng đang chạy tại http://localhost:${port}`);

});

***BT* Web back-end (6) - Cấu hình Web server trong Nodejs**

Câu 6.2: Làm sao biết gói được cài local hay global? Và chúng được lưu ở đâu?

1. Nhận biết gói cài local hay global:

* Cài local:  
  Gói sẽ được cài trong thư mục node\_modules của dự án hiện tại.



* Cài global:  
  Gói sẽ được cài toàn cục và có thể dùng ở mọi nơi trong máy.  
  

2. Các gói được lưu ở đâu?

* Local:  
  Thư mục: ./node\_modules (cùng cấp với package.json của dự án)
* Global:  
  Dùng lệnh để biết đường dẫn:



3. Khi nào dùng cài global?  
- Khi muốn dùng công cụ/dòng lệnh ở mọi nơi (VD: nodemon, http-server, typescript...)

Câu 6.3: Sự khác nhau giữa dependencies và devDependencies

| dependencies | devDependencies |
| --- | --- |
| Là những gói cần khi chạy ứng dụng thực tế | Là những gói chỉ dùng khi phát triển, không cần khi ứng dụng chạy thật |
| Ví dụ: express, mongoose | Ví dụ: nodemon, eslint, jest |
| Cài bằng: npm install package-name | Cài bằng: npm install --save-dev package-name |

Câu 6.4: package.json dùng để làm gì? Phát biểu nào KHÔNG đúng?

A. Đúng – chứa tên app, version, license, v.v.

B. Đúng – quản lý dependencies/devDependencies

C. Đúng – có thể chứa "main": "index.js" → chỉ ra file JS đầu tiên chạy

D. Sai – Không chứa mã giao diện (UI), mà UI thường nằm trong các file .html, .js, .css riêng biệt.

Đáp án: D

Câu 6.5: PNPM là gì? Phát biểu nào KHÔNG đúng?

A. Đúng – PNPM dùng hard link nên tiết kiệm dung lượng ổ đĩa.

B. Đúng – Có thể dùng thay NPM (hoặc song song).

C. Đúng – PNPM là viết tắt của Performant Node Package Manager

D. Sai – PNPM có thể thay thế NPM.

Câu 6.6: Express trong Node.js dùng để làm gì?

Đáp án :A. Tạo ứng dụng web và API

Giải thích:

Express là web framework nhẹ cho Node.js, dùng để:

Tạo web server

Xử lý API

Quản lý router, middleware

Câu 6.7: Nodemon trong Node.js dùng để làm gì?

Đáp án: D. Tự động khởi động lại server khi mã thay đổi

Giải thích:

Nodemon giám sát file .js, khi có thay đổi thì khởi động lại app tự động, không cần dừng thủ công và chạy lại.

**Web back-end (7) - Local và Global, Dependencies và devDependencies**

1. Khái niệm Local và Global trong NPM

* Cài đặt cục bộ (Local): Gói được cài đặt chỉ áp dụng trong phạm vi dự án hiện tại. Thông tin về gói này được lưu trong tập tin package.json dưới mục dependencies, và mã nguồn của gói được đặt trong thư mục node\_modules của dự án.​
* Cài đặt toàn cục (Global): Gói được cài đặt có thể sử dụng trên toàn hệ thống, không phụ thuộc vào dự án cụ thể nào. Khi cài đặt theo kiểu này, thông tin về gói không được ghi vào package.json của bất kỳ dự án nào

2. Cài đặt gói theo kiểu cục bộ

* Cú pháp lệnh: npm install <package\_name>

3. Cài đặt gói theo kiểu toàn cục

* Cú pháp lệnh: npm install <package\_name> -g

4. Phân biệt Dependencies và devDependencies

* Dependencies: Là các gói cần thiết cho ứng dụng hoạt động bình thường. Chúng được liệt kê trong mục dependencies của package.json.​
* devDependencies: Là các gói chỉ cần thiết trong quá trình phát triển, như các công cụ kiểm thử hoặc biên dịch. Chúng được liệt kê trong mục devDependencies của package.json.

***BT Web back-end (7) - Local và Global, Dependencies và devDependencies***

Câu 7.2:

Câu hỏi: Trong môi trường phát triển ứng dụng Node.js, phát biểu nào không đúng khi nói về gói cục bộ?

A. Gói cục bộ được cài đặt trong thư mục node\_modules của dự án  
Đúng – Khi bạn cài đặt gói mà không có cờ -g, ví dụ npm install express, thì nó được cài vào node\_modules trong thư mục dự án.

B. Gói cục bộ chỉ có thể được sử dụng trong dự án mà nó được cài đặt  
Đúng – Mỗi dự án có thư viện riêng biệt; các dự án khác không thể dùng gói đó nếu chưa cài riêng.

C. Gói cục bộ được quản lý thông qua file package.json và có thể sử dụng các phiên bản khác nhau giữa các dự án  
Đúng – Mỗi dự án có thể khai báo và dùng phiên bản riêng biệt của gói trong package.json.

D. Gói cục bộ luôn được cài đặt toàn cục trên hệ thống để tất cả các dự án đều có thể truy cập  
Sai – Đây là mô tả của gói toàn cục, không phải gói cục bộ.

Đáp án: D

Câu 7.3:

Câu hỏi: Trong môi trường phát triển ứng dụng Node.js, phát biểu nào không đúng khi nói về gói toàn cục?

A. Gói toàn cục luôn được liệt kê trong tập tin package.json của dự án  
Sai – Gói toàn cục cài bằng npm install -g sẽ không được liệt kê trong package.json của bất kỳ dự án nào.

B. Gói toàn cục thường được sử dụng cho các công cụ dòng lệnh (CLI) như nodemon  
Đúng – Các gói như nodemon, eslint, npm thường được cài toàn cục để dùng mọi nơi.

C. Gói toàn cục được lưu trong thư mục toàn cục của hệ thống và có thể truy cập từ bất kỳ dự án nào  
Đúng – Gói toàn cục dùng được ở bất kỳ đâu trong máy.

D. Gói toàn cục được cài đặt bằng lệnh npm install -g <package-name>  
Đúng – Dấu -g là viết tắt của "global" – để cài toàn cục.

Đáp án: A

Câu 7.4:

Câu hỏi: Trong môi trường phát triển ứng dụng Node.js, phát biểu nào không đúng khi nói về kiểu cài đặt dependencies?

A. Các gói trong dependencies được cài đặt cục bộ trong dự án bằng lệnh npm install <package-name>  
Đúng – Khi dùng npm install express, gói sẽ vào dependencies.

B. Các gói trong dependencies là các phụ thuộc cần thiết để ứng dụng chạy trong môi trường triển khai, sản xuất (production)  
Đúng – Ví dụ: express, axios, mongoose,… là những thư viện cần để app hoạt động.

C. Các gói trong dependencies chỉ được sử dụng trong giai đoạn phát triển và không cần thiết khi triển khai ứng dụng (production)  
Sai – Đây là vai trò của devDependencies, còn dependencies là cần cho production.

D. Các gói trong dependencies được liệt kê trong tập tin package.json và tự động cài đặt khi chạy npm install  
Đúng – Khi chạy npm install, tất cả các gói trong dependencies sẽ được cài.

Đáp án: C

Câu 7.5:

Câu hỏi: Trong môi trường phát triển ứng dụng Node.js, phát biểu nào không đúng khi nói về kiểu cài đặt devDependencies?

A. Các gói trong devDependencies được cài đặt bằng lệnh npm install <package-name> --save-dev  
Đúng – Dùng cờ --save-dev để thêm vào devDependencies.

B. Các gói trong devDependencies là các phụ thuộc cốt lõi để ứng dụng chạy trong môi trường triển khai (production)  
Sai – devDependencies là để dùng trong phát triển, không cần khi chạy production.

C. Các gói trong devDependencies chỉ cần thiết trong giai đoạn phát triển hoặc kiểm thử, không cần cho môi trường triển khai (production)  
Đúng – Ví dụ: nodemon, eslint, jest là công cụ dùng khi dev/test.

D. Các gói trong devDependencies không được cài đặt khi chạy npm install --production  
Đúng – Lệnh đó sẽ bỏ qua devDependencies.

Đáp án: B

**Web back-end (8) - Một số chủ đề JavaScript (1)**

1. Lập trình đồng bộ trong JavaScript

Lập trình đồng bộ (synchronous programming) trong JavaScript là kỹ thuật mà các lệnh được thực thi tuần tự, từ trên xuống dưới. Mỗi lệnh phải hoàn thành trước khi lệnh tiếp theo được thực thi. Điều này có nghĩa là nếu một tác vụ mất nhiều thời gian (như đọc tập tin, gọi API), chương trình sẽ bị "chặn" cho đến khi tác vụ đó hoàn tất.​

Ví dụ về lập trình đồng bộ:

function cong(a, b) {

return a + b;

}

console.log("Bắt đầu");

let kq = cong(6, 8); // thực thi đồng bộ

console.log("Kết quả:", kq);

console.log("Kết thúc");

Kết quả khi chạy:

Bắt đầu

Kết quả: 14

Kết thúc

2. Hàm ẩn danh trong JavaScript

Hàm ẩn danh (anonymous function) là hàm không có tên khi được định nghĩa. Thay vì khai báo với từ khóa function đi kèm tên, hàm ẩn danh thường được gán trực tiếp vào biến, truyền làm tham số, hoặc thực thi ngay lập tức mà không cần gọi hàm.

VD:

// Gán vào biến

const cong = function(a, b) {

return a + b;

};

// Truyền làm tham số

setTimeout(function() {

console.log("Đã hết 3 giây!");

}, 3000);

// Thực thi ngay lập tức

(function() {

console.log("Hàm chạy ngay sau khi định nghĩa!");

})();

***BT Web back-end (8) - Một số chủ đề JavaScript (1)***

Câu 8.2:

Câu hỏi: Lập trình đồng bộ trong JavaScript là gì? Phát biểu nào sau đây không đúng?

A. Lập trình đồng bộ không chặn luồng, cho phép các lệnh tiếp theo chạy ngay cả khi tác vụ chưa hoàn tất  
Sai – Đây là mô tả của lập trình bất đồng bộ (asynchronous).  
→ Trong lập trình đồng bộ (synchronous), các lệnh sẽ chặn luồng, nghĩa là lệnh sau phải đợi lệnh trước hoàn thành.

B. Các lệnh được thực thi tuần tự, theo thứ tự từ trên xuống dưới  
Đúng – Đây là đặc trưng cơ bản của synchronous: thực thi dòng lệnh từ trên xuống dưới, từng lệnh một.

C. Mỗi lệnh phải hoàn thành trước khi lệnh tiếp theo được thực thi  
Đúng – Đồng bộ = chờ đợi. Lệnh tiếp theo không chạy cho tới khi lệnh hiện tại hoàn tất.

D. Phù hợp với các tác vụ đơn giản như tính toán cơ bản không cần chờ đợi  
Đúng – Với những tác vụ đơn giản, không cần thao tác với file, mạng,... thì lập trình đồng bộ vẫn hiệu quả.

Đáp án: A – Vì đây là phát biểu không đúng về lập trình đồng bộ.

Câu 8.3:

Câu hỏi: Hàm ẩn danh (anonymous function) trong JavaScript là gì? Phát biểu nào sau đây không đúng?

A. Hàm ẩn danh có thể truy cập biến trong phạm vi bao quanh nhờ closure  
Đúng – Đây là một trong những điểm mạnh của JS: closures cho phép hàm (kể cả ẩn danh) truy cập biến ngoài phạm vi của nó.

B. Hàm ẩn danh là hàm không có tên khi được định nghĩa  
Đúng – Ví dụ:

function () {} // lỗi cú pháp vì không gán, nhưng đúng định nghĩa là "hàm không tên"

const sayHi = function () { console.log("Hi"); }; // ✔ hợp lệ

C. Hàm ẩn danh thường được gán vào biến hoặc truyền làm tham số cho hàm khác  
Đúng – Đây là cách thường dùng nhất của hàm ẩn danh, đặc biệt với callback:

setTimeout(function() { console.log("Done"); }, 1000);

D. Hàm ẩn danh được "nâng lên" (hoisted) giống như hàm khai báo (function declaration)  
Sai – Chỉ có function declaration mới được hoisted (nâng lên đầu scope).  
→ Hàm ẩn danh gán vào biến thì không hoisted hoặc chỉ hoisted phần khai báo biến, không hoisted phần gán hàm.

sayHello(); //lỗi: Cannot access 'sayHello' before initialization

const sayHello = function () {

console.log("Hello");

};

Đáp án: D

**Web back-end (9) - Một số chủ đề JavaScript (2)**

1. Hàm mũi tên (Arrow Function)

Hàm mũi tên, được giới thiệu trong ES6 (ECMAScript 2015), cung cấp một cách viết hàm ngắn gọn và hiện đại trong JavaScript. Chúng thường được sử dụng trong lập trình bất đồng bộ và các tình huống khác. Hàm mũi tên sử dụng ký hiệu => để kết nối tham số và thân hàm, mang lại sự ngắn gọn và một số đặc tính độc đáo về ngữ nghĩa.​

*Cú pháp của hàm mũi tên:*

* Không tham số: () => { ... }
* Một tham số: x => { ... } (không cần dấu ngoặc cho một tham số)​
* Nhiều tham số: (x, y) => { ... }

Hàm không có tham số:

const thongBao = () => {

console.log("Chưa có thông báo, khi nào có thông báo sẽ thông báo!");

}

thongBao();

Hàm có một tham số:

const binhPhuong = x => x \* x;

console.log(binhPhuong(3)); // 9

Hàm có nhiều tham số:

const cong = (x, y) => x + y;

console.log(cong(2, 4)); // 6

2. Lập trình bất đồng bộ trong JavaScript

Lập trình bất đồng bộ (asynchronous programming) cho phép JavaScript thực thi các tác vụ mà không chặn luồng chính, giúp cải thiện hiệu suất và trải nghiệm người dùng. Trong lập trình đồng bộ, các lệnh được thực thi tuần tự, và một lệnh phải hoàn thành trước khi lệnh tiếp theo được thực thi. Điều này có thể gây ra tình trạng "chặn" khi gặp các tác vụ mất nhiều thời gian như đọc tập tin, gọi API, hoặc giao tiếp client-server.

***BT Web back-end (9) - Một số chủ đề JavaScript (2)***

Câu 9.2:

Câu hỏi: Hàm mũi tên (arrow function) là gì? Phát biểu nào sau đây không đúng về hàm mũi tên?

A. Hàm mũi tên có cú pháp ngắn gọn, giúp viết mã dễ đọc hơn  
Đúng – Đây là lý do phổ biến khiến arrow function được ưa chuộng.

// Hàm thường:

function add(a, b) { return a + b; }

// Hàm mũi tên:

const add = (a, b) => a + b;

B. Hàm mũi tên có this (ngữ cảnh) riêng, không kế thừa từ phạm vi bên ngoài  
Sai – Arrow function KHÔNG có ngữ cảnh this riêng, mà kế thừa this từ phạm vi bên ngoài (lexical scope).  
→ Đây là điểm quan trọng nhất phân biệt arrow function với function thường.

C. Hàm mũi tên không thể được dùng làm hàm tạo (constructor)  
Đúng – Bạn không thể dùng new với arrow function. Thử sẽ gây lỗi.

D. Hàm mũi tên không có biến arguments để truy cập danh sách tham số  
Đúng – Arrow function không có arguments, bạn phải dùng rest parameter (...args) nếu muốn lấy tất cả tham số.

Đáp án: B – vì đây là phát biểu sai về arrow function.

Câu 9.3:

Câu hỏi: Lập trình bất đồng bộ là gì? Phát biểu nào sau đây không đúng về lập trình bất đồng bộ?

A. Lập trình bất đồng bộ cho phép chương trình thực hiện nhiều tác vụ cùng lúc mà không cần chờ đợi tác vụ trước đó hoàn thành  
Đúng – Đó là mục tiêu chính của bất đồng bộ: không chặn luồng chính.

B. Lập trình bất đồng bộ đặc biệt hữu ích khi xử lý các tác vụ tốn thời gian như yêu cầu mạng, đọc/ghi tập tin hoặc tương tác với cơ sở dữ liệu  
Đúng – Những thao tác I/O (Input/Output) thường mất thời gian nên async là cần thiết.

C. Callback là một kỹ thuật bất đồng bộ, nó sẽ làm cho code dễ đọc và dễ bảo trì hơn khi code có nhiều callback lồng nhau  
Sai – Callback lồng nhau thường gây ra cái gọi là "callback hell", khiến code khó đọc và khó bảo trì.  
→ Vì vậy, phát biểu này không đúng.

D. Async/await cho phép viết mã bất đồng bộ trông giống như mã đồng bộ, giúp mã trở nên dễ đọc và dễ hiểu hơn  
Đúng – Đây là ưu điểm lớn nhất của async/await.

Đáp án: C – vì callback lồng nhau không giúp dễ đọc, ngược lại còn làm rối code.

**Web back-end (10) - Hàm callback**

1. Hàm callback là gì?

Trong JavaScript, hàm callback là một hàm được truyền vào một hàm khác như một tham số và sẽ được thực thi sau khi hàm "cha" hoàn thành một tác vụ nào đó. Nói cách khác, callback cho phép một đoạn mã được thực thi sau khi một đoạn mã khác hoàn thành. Đây là một trong ba kỹ thuật được sử dụng để lập trình bất đồng bộ trong JavaScript.​

2. Đặc điểm của hàm callback:

* Truyền như tham số: Callback là một hàm được truyền vào hàm khác để thực thi sau khi một điều kiện hoặc tác vụ hoàn tất
* Thực thi bất đồng bộ: Thường dùng để xử lý các tác vụ không chặn luồng chính (non-blocking), như truy cập thiết bị (đọc/ghi tập tin), gọi API, hoặc trong các hàm hẹn giờ (timer).​
* Tính linh hoạt: Có thể là hàm ẩn danh (anonymous function), hàm mũi tên (arrow function), hoặc hàm đã được định nghĩa trước (function expression)
* "Callback Hell": Nếu lồng quá nhiều callback, mã nguồn có thể trở nên khó đọc và khó bảo trì. Callback hell thường được giải quyết bằng Promise hoặc async/await.

3. Ứng dụng của hàm callback:

Xử lý sự kiện (event handling)

Tác vụ bất đồng bộ

Xử lý mảng

Hoàn thành tác vụ

Tùy chỉnh logic

***BT Web back-end (10) - Hàm callback***

Câu 10.2

Câu hỏi: Hàm callback trong JavaScript là gì? Phát biểu nào sau đây không đúng?

A. Hàm callback là một hàm được truyền vào một hàm khác như một tham số  
Đúng – Đây là định nghĩa chính xác nhất của callback.

B. Hàm callback thường được sử dụng để xử lý các tác vụ bất đồng bộ  
Đúng – Callback được dùng nhiều trong bất đồng bộ như setTimeout, gọi API,...

C. Hàm callback có thể là hàm ẩn danh, hàm mũi tên hoặc hàm đã được định nghĩa trước  
Đúng – Callback có thể có nhiều hình thức, ví dụ:

setTimeout(function () { ... }); // ẩn danh

setTimeout(() => { ... }); // arrow function

setTimeout(myFunction, 1000); // hàm định nghĩa trước

D. Hàm callback được thực thi ngay lập tức sau khi hàm "cha" bắt đầu thực hiện  
Sai – Callback không được thực thi ngay lập tức, mà thường được gọi lại (gọi sau) khi một sự kiện xảy ra hoặc tác vụ hoàn tất.

console.log("Start");

setTimeout(() => {

console.log("Callback executed"); // Gọi sau

}, 1000);

console.log("End");

Đáp án: D – vì đây là phát biểu không đúng về callback.

Câu 10.3

Câu hỏi: Callback hell trong JavaScript là gì? Phát biểu nào sau đây không đúng?

A. Callback hell xảy ra khi có quá nhiều hàm callback lồng nhau, khiến mã nguồn khó đọc và bảo trì  
Đúng – Đây là mô tả chuẩn xác nhất về callback hell.

B. Callback hell thường xuất hiện khi xử lý các tác vụ bất đồng bộ phức tạp  
Đúng – Khi bạn phải gọi API nối tiếp, thao tác I/O nhiều lần,... dễ rơi vào callback hell.

C. Callback hell là một cách hiệu quả để quản lý các tác vụ bất đồng bộ trong JavaScript  
Sai – Callback hell không hiệu quả. Nó là vấn đề cần tránh, không phải phương pháp hay.

D. Callback hell có thể được giải quyết bằng Promise hoặc async/await  
Đúng – Promise và async/await ra đời để giúp code dễ đọc, thoát khỏi callback hell.

Đáp án: C – vì callback hell không phải là cách hiệu quả, ngược lại là nỗi ám ảnh của dev.

**Git thực hành (3) - Cấu hình định danh người dùng**

System (hệ thống): Áp dụng cho tất cả người dùng và kho lưu trữ trên hệ thống. Tập tin cấu hình nằm tại C:\Program Files\Git\etc\gitconfig trên Windows. Cần quyền quản trị để thiết lập.​

Global (người dùng): Áp dụng cho tất cả kho lưu trữ của người dùng hiện tại. Tập tin cấu hình nằm tại C:\Users\<tên\_người\_dùng>\.gitconfig trên Windows. Đây là phạm vi được sử dụng phổ biến nhất.​

Local (kho lưu trữ): Áp dụng cho một kho lưu trữ cụ thể. Tập tin cấu hình nằm tại .git\config trong thư mục gốc của kho lưu trữ.

Để thiết lập tên và email cho tất cả các kho lưu trữ của người dùng hiện tại, sử dụng lệnh:​

git config --global user.name "Tên của bạn"

git config --global user.email [email@example.com](mailto:email@example.com)

Để kiểm tra các cấu hình đã thiết lập, dùng lệnh:

git config –list

***BT GIT THỰC HÀNH (3) - CẤU HÌNH ĐỊNH DANH NGƯỜI DÙNG***

Câu 3.2: Phạm vi cấu hình định danh trong Git – Phát biểu nào không đúng?

A. Phạm vi system áp dụng cho tất cả người dùng và kho lưu trữ trên hệ thống  
Đúng – Đây là cấu hình hệ thống toàn cục.

B. Phạm vi global áp dụng cho tài khoản người dùng hiện tại và tất cả kho lưu trữ mà người dùng đó làm việc  
Đúng – Đây là cấu hình phổ biến nhất (dùng --global).

C. Phạm vi local áp dụng cho một kho lưu trữ cụ thể và có độ ưu tiên thấp nhất  
Sai – Local có độ ưu tiên cao nhất, vì Git sẽ ưu tiên đọc config theo thứ tự:

local > global > system

D. Phạm vi local áp dụng cho một kho lưu trữ cụ thể và có độ ưu tiên cao nhất  
Đúng – Đây là câu mô tả chính xác.

Đáp án: C – vì phát biểu này không đúng.

*Nếu muốn kiểm tra lại cấu hình hiện tại trong từng thư mục:*

git config user.name

git config user.email

*Hoặc xem toàn bộ config của repo:*

git config –list

**Git thực hành (4) - Các khu vực làm việc của Git**

Thư mục làm việc (Working Directory): Đây là nơi bạn trực tiếp chỉnh sửa, thêm hoặc xóa các tập tin trên máy tính của mình. Nó chứa bản sao hiện tại của dự án, bao gồm cả các thay đổi chưa được theo dõi bởi Git. Ví dụ, khi bạn tạo một tập tin mới như index.js trong dự án, tập tin này sẽ xuất hiện trong thư mục làm việc.​

Khu vực tổ chức tạm (Staging Area): Còn gọi là khu tạm, đây là vùng trung gian lưu trữ các thay đổi mà bạn đã chọn để chuẩn bị cho việc commit. Các thay đổi được thêm vào khu tạm bằng lệnh git add. Ví dụ, sau khi chỉnh sửa index.js, bạn sử dụng git add index.js để đưa thay đổi này vào khu tạm.​

Kho chứa (Repository): Đây là nơi lưu trữ tất cả các tập tin và lịch sử thay đổi của dự án. Thư mục .git trong dự án chính là kho chứa. Khi bạn thực hiện lệnh git commit, các thay đổi từ khu tạm sẽ được ghi vào kho chứa, tạo thành một điểm lưu trữ trong lịch sử phiên bản của dự án.

***BT GIT THỰC HÀNH (4) - CẤU HÌNH ĐỊNH DANH NGƯỜI DÙNG***

Câu 4.2: Ba khu vực làm việc chính của Git – Phát biểu nào không đúng?

A. Thư mục làm việc là nơi bạn chỉnh sửa tập tin trực tiếp.  
Đúng – Đây là nơi bạn thao tác trên file thực tế.

B. Khu tạm là nơi lưu trữ lịch sử phiên bản của dự án.  
Sai – Lưu trữ lịch sử phiên bản là vai trò của Repository, không phải Staging Area.

C. Kho chứa là nơi lưu trữ dữ liệu chính thức của dự án.  
Đúng – Repository chứa toàn bộ commit, là dữ liệu chính thức.

D. Lệnh git add được sử dụng để chuyển các thay đổi từ Thư mục làm việc sang Khu tạm.  
Đúng – Chính xác, git add dùng để đưa file từ Working Directory → Staging Area.

Đáp án: B – vì phát biểu không đúng về Staging Area.

Câu 4.3: Thư mục làm việc trong Git – Phát biểu nào không đúng?

A. Là thư mục thực tế trên máy tính, nơi bạn làm việc trực tiếp với các tập tin của dự án.  
Đúng – Đây là định nghĩa chuẩn của working directory.

B. Là nơi bạn tạo, sửa, hoặc xóa tập tin trước khi đưa thay đổi vào .git.  
Đúng – Bạn thao tác file tại đây, trước khi commit.

C. Là nơi lưu trữ lịch sử phiên bản và dữ liệu chính thức của dự án.  
Sai – Vai trò này thuộc về Repository (.git), không phải thư mục làm việc.

D. Phản ánh các thay đổi so với commit cuối cùng, giúp bạn quyết định những gì cần thêm vào staging area.  
Đúng – Thư mục làm việc cho bạn thấy sự khác biệt so với repo, dùng git status để kiểm tra.

Đáp án: C – vì nó mô tả sai vai trò của thư mục làm việc.

**Git thực hành (5) - Các khu vực làm việc của Git**

1. Khu tạm (Staging Area):

Khu tạm, còn được gọi là index, là một khu vực trung gian giữa Thư mục làm việc (Working Directory) và Kho chứa (Repository). Nó cho phép bạn chọn lọc và tổ chức các thay đổi mà bạn muốn đưa vào commit tiếp theo.​[Langbiang](https://legiacong.blogspot.com/2025/03/git-thuc-hanh-5-cac-khu-vuc-lam-viec.html?m=1&utm_source=chatgpt.com)

* Gỡ bỏ tập tin khỏi Khu tạm: Nếu bạn đã thêm một tập tin vào Khu tạm bằng lệnh git add nhưng sau đó quyết định không muốn commit tập tin đó nữa, bạn có thể sử dụng lệnh git rm --cached <tên\_tập\_tin> để gỡ bỏ tập tin khỏi Khu tạm mà không xóa nó khỏi Thư mục làm việc. Ví dụ:

git rm --cached index.js

2. Tập tin .gitignore:

Tập tin .gitignore là một cơ chế trong Git cho phép bạn chỉ định các tập tin hoặc thư mục mà Git sẽ bỏ qua khi thực hiện các lệnh như git add hoặc git status. Điều này giúp loại bỏ các tập tin không cần thiết khỏi lịch sử phiên bản, như tập tin tạm, tập tin log, hoặc thư viện phụ thuộc.​

* Công dụng của .gitignore:
  + Tránh commit "rác": Ngăn chặn việc thêm các tập tin không liên quan vào kho chứa.​
  + Giữ lịch sử sạch sẽ: Tập trung vào mã nguồn chính, giúp lịch sử commit rõ ràng và dễ theo dõi.​
  + Tăng hiệu quả: Giảm kích thước kho chứa và tránh xung đột không cần thiết.​
* Cách sử dụng .gitignore:
  + Tạo tập tin .gitignore trong thư mục gốc của dự án.​
  + Thêm các mẫu (pattern) của tập tin hoặc thư mục muốn bỏ qua vào tập tin .gitignore. Ví dụ

node\_modules/

dist/

.env

\*.log

\*.tmp

***BT GIT THỰC HÀNH (5) - CẤU HÌNH ĐỊNH DANH NGƯỜI DÙNG***

Câu 5.2: Khu tạm (staging area) trong Git – Phát biểu nào sau đây không đúng?

Khu tạm (staging area) là nơi bạn chuẩn bị các thay đổi trước khi commit vào kho chứa. Đây là bước trung gian giữa working directory và repository.

A. Khu tạm là một khu vực trung gian giữa Thư mục làm việc (working directory) và Kho chứa (repository).  
Đúng – Đây là mô tả chính xác về vai trò của khu tạm trong Git.

B. Khu tạm là một thư mục vật lý trong Thư mục làm việc.  
Sai – Khu tạm không phải là một thư mục vật lý. Nó chỉ là một không gian quản lý trong Git để lưu trữ các thay đổi đã được đánh dấu trước khi commit.

C. Khu tạm cho phép bạn chọn lọc và tổ chức các thay đổi bạn muốn đưa vào commit tiếp theo.  
Đúng – Khu tạm cho phép bạn lựa chọn các thay đổi cụ thể để đưa vào commit tiếp theo, giúp quản lý mã nguồn tốt hơn.

D. Lệnh git rm --cached <tên\_tập\_tin> được sử dụng để gỡ bỏ tên tập tin ra khỏi Khu tạm.  
Đúng – Lệnh này giúp bạn xóa một file khỏi staging area mà không xóa nó khỏi thư mục làm việc.

Đáp án: B – vì phát biểu không đúng về khu tạm trong Git.

Câu 5.3: gitignore là gì? Phát biểu nào sau đây không đúng?

gitignore là một cơ chế trong Git để bỏ qua các file không cần thiết (ví dụ như file log, thư mục build, hay thư viện phụ thuộc như node\_modules).

A. gitignore giúp thêm các tập tin không liên quan (ví dụ: node\_modules, tập tin log) vào kho chứa.  
Sai – gitignore không thêm các file này vào kho chứa. Mục đích của gitignore là để bỏ qua các file không cần thiết, không đưa chúng vào kho chứa.

B. gitignore là một cơ chế trong Git, cho phép bạn chỉ định các tập tin hoặc thư mục mà Git sẽ bỏ qua khi thực hiện các lệnh như git add hoặc git status.  
Đúng – Đây là mô tả đúng về cách gitignore hoạt động.

C. gitignore được thực hiện thông qua tập tin có tên là .gitignore, thường đặt ở thư mục gốc của dự án.  
Đúng – File .gitignore chứa các mẫu (patterns) của các file hoặc thư mục mà Git sẽ bỏ qua.

D. Mục đích của gitignore là loại bỏ các tập tin không cần thiết khỏi lịch sử phiên bản, như tập tin tạm, tập tin nhị phân, hoặc thư viện phụ thuộc.  
Đúng – Mục đích của .gitignore là không thêm các file không cần thiết vào lịch sử phiên bản của Git.

Đáp án: A – vì phát biểu này không đúng về mục đích của .gitignore.

Câu 5.4: Kho chứa (repository) trong Git – Phát biểu nào sau đây không đúng?

Kho chứa (repository) là nơi Git lưu trữ tất cả các commit và lịch sử thay đổi của dự án.

A. Kho chứa là nơi lưu trữ tất cả các tập tin và lịch sử thay đổi của một dự án.  
Đúng – Đây là mô tả chính xác về kho chứa trong Git.

B. Thư mục .git trong dự án chính là Kho chứa.  
Đúng – Thư mục .git chứa toàn bộ thông tin quản lý của Git, đây là local repository của bạn.

C. Lệnh git commit được sử dụng để chuyển tập tin từ Thư mục làm việc (working directory) sang Kho chứa.  
Đúng – Lệnh git commit ghi lại các thay đổi trong kho chứa từ khu tạm (staging area).

D. Kho chứa cục bộ (local repository) là kho chứa được lưu trữ trên một máy chủ ở xa.  
Sai – Kho chứa cục bộ (local repository) là kho chứa nằm trên máy tính của bạn. Kho chứa từ xa (remote repository) là kho chứa trên máy chủ từ xa (ví dụ GitHub, GitLab,...).

Đáp án: D – vì phát biểu này không đúng về kho chứa cục bộ.

Câu 5.5: Commit trong Git là gì? Phát biểu nào sau đây không đúng?

Commit trong Git là hành động lưu lại các thay đổi vào kho chứa, và mỗi commit đều có một thông điệp mô tả sự thay đổi.

A. Commit là lệnh trong Git dùng để lưu các thay đổi từ Khu tạm vào Kho chứa.  
Đúng – Lệnh git commit dùng để đưa các thay đổi từ staging area vào kho chứa.

B. Commit tạo ra một "ảnh chụp" (snapshot) mới trong lịch sử phiên bản.  
Đúng – Commit là một bản sao (snapshot) của tất cả các file đã được thay đổi trong kho chứa tại thời điểm đó.

C. Commit dùng để chuyển tập tin từ Kho chứa sang Khu tạm.  
Sai – Lệnh git add mới là lệnh đưa file từ Working Directory vào Staging Area (Khu tạm). Commit chuyển từ Staging Area vào Kho chứa.

D. Mỗi commit đi kèm một thông điệp (message) mô tả thay đổi.  
Đúng – Mỗi commit cần có một thông điệp để mô tả những thay đổi đã được thực hiện.

Đáp án: C – vì commit không dùng để chuyển file từ kho chứa sang khu tạm.

**Web back-end (11) - Lập trình giao diện**

11. Lập trình giao diện

11.1 Commit đầu tiên

* Sau khi học về JavaScript và cài đặt môi trường Express (cùng nodemon), bạn cần quản lý mã nguồn bằng Git.
* Tạo file .gitignore trong thư mục dự án TeoShop, có thể sao chép nội dung từ: 👉 <https://github.com/github/gitignore/blob/main/Node.gitignore>
* Thực hiện commit đầu tiên:

git add .

git commit -m "Initialize TeoShop project with Express and Nodemon"

git log

* Mục tiêu: lưu lại trạng thái ban đầu của dự án trong hệ thống quản lý phiên bản Git.

11.2 Sử dụng Free HTML Template

Free HTML Template là gì?

* Giao diện web được thiết kế sẵn bằng HTML, CSS, JS, miễn phí từ các nguồn như BootstrapMade, htmlcodex,...
* Cấu trúc chuẩn: header, footer, sidebar, v.v.

Khi nào nên dùng?

* Dự án nhỏ, cần giao diện nhanh, ngân sách thấp, hoặc để làm prototype trình bày ý tưởng.

Cách sử dụng:

1. Tải template từ nguồn đáng tin cậy.
2. Giải nén và copy vào thư mục dự án (đổi tên thành public).
3. Tuỳ chỉnh nội dung: văn bản, hình ảnh, CSS...
4. Tối ưu mã (nén CSS/JS, kiểm tra responsive...).

Ưu điểm:

* Nhanh chóng, chuyên nghiệp, dễ học hỏi.

Nhược điểm:

* Khó tùy biến hoàn toàn, có thể dư mã, trùng lặp giao diện.

11.3 Static files trong Express

* Static files: tập tin tĩnh không cần xử lý server-side (HTML, CSS, JS, ảnh...).
* Thường đặt trong thư mục public.

Cấu hình để Express phục vụ static files:

Trong file index.js, thêm:

app.use(express.static(\_\_dirname + '/public'));

Giải thích:

* app.use(): đăng ký middleware.
* express.static(): middleware phục vụ file tĩnh.
* \_\_dirname: đường dẫn thư mục hiện tại.
* /public: thư mục chứa giao diện tĩnh.

Kết quả:

* Khi chạy web (<http://localhost:9000/>), trình duyệt sẽ hiển thị trang giao diện từ template đã đặt trong thư mục public.

***BT Web back-end (11) - Lập trình giao diện***

Câu 11.2: Free HTML template là gì? Phát biểu nào sau đây không đúng?

Free HTML template là các mẫu giao diện website có sẵn, được thiết kế bằng HTML, CSS (và có thể có JavaScript) để giúp bạn tạo giao diện nhanh hơn.

A. Free HTML template là các mẫu giao diện web được thiết kế sẵn, bao gồm HTML, CSS, và đôi khi JavaScript, được cung cấp miễn phí.  
Đúng – Đây là định nghĩa chuẩn xác.

B. Free HTML template thường có giao diện đẹp, cấu trúc cơ bản (header, footer, sidebar), và phù hợp cho nhiều loại dự án như blog, portfolio, hoặc trang doanh nghiệp nhỏ.  
Đúng – Đa số các template đều hỗ trợ những thành phần giao diện cơ bản này.

C. Sử dụng Free HTML template tốn nhiều thời gian, phù hợp cho các dự án lớn.  
Sai – Ngược lại, mục tiêu của free template là tiết kiệm thời gian, phù hợp với dự án nhỏ hoặc vừa, giúp bạn khởi đầu nhanh chóng, chứ không dành cho dự án lớn cần tùy biến sâu.

D. Free HTML template cho phép bạn tùy chỉnh CSS, thay đổi nội dung (văn bản, hình ảnh), tích hợp back-end (như Node.js, PHP).  
Đúng – Dù là "template tĩnh", bạn vẫn có thể dùng để kết hợp với back-end, nếu cần.

Đáp án: C – vì phát biểu này không đúng với đặc điểm của Free HTML template.

Câu 11.3: Static files trong Express là gì? Phát biểu nào sau đây không đúng?

Trong Express, static files là những tệp như hình ảnh, CSS, JS... mà không cần xử lý thêm từ phía server, chỉ đơn giản là gửi trực tiếp cho client.

A. Static files là các tập tin không thay đổi nội dung hoặc không cần xử lý logic phía server.  
Đúng – Chúng chỉ được “serve” lên client mà không cần tính toán gì cả.

B. Static files được gửi trực tiếp cho client thông qua middleware express.static.  
Đúng – Đây là cách phổ biến để cung cấp file tĩnh trong Express.

C. Mục đích của static files là tối ưu hóa hiệu suất, tổ chức dự án và bảo mật.  
Đúng – Việc tách static files giúp tổ chức mã rõ ràng và tăng hiệu suất khi trình duyệt cache.

D. Không nên đặt các static files trong thư mục public.  
Sai – Thực tế public là thư mục mặc định để chứa static files trong nhiều dự án Express. Đây là thực hành phổ biến.

Đáp án: D – vì phát biểu không đúng về cách dùng static files trong Express.

**Web back-end (12) - Ứng dụng client-server**

12. Ứng dụng client-server

12.1 Ứng dụng kiểu cục bộ (Local Application)

Khái niệm:

* Chạy trực tiếp trên thiết bị của người dùng, không cần kết nối liên tục với Internet.

Đặc điểm:

* Chạy độc lập: không phụ thuộc vào mạng.
* Truy cập tài nguyên thiết bị: như hệ thống tập tin, phần cứng.
* Hiệu suất cao: do chạy trên máy cục bộ.
* Hoạt động ngoại tuyến.
* Bảo mật tốt hơn: dữ liệu lưu trên thiết bị.

Ví dụ:

* Microsoft Word, Excel, Photoshop, Notepad, các game offline.

Lập trình ứng dụng cục bộ:

* Sử dụng các ngôn ngữ như C, C++, C#, Java, Python...
* Có hàm main() làm điểm bắt đầu.
* Xuất hiện menu chính hoặc cửa sổ chính, đợi tương tác người dùng.
* Sử dụng trình dịch cục bộ để biên dịch và chạy.
* Có thể dùng thư viện và cơ sở dữ liệu cục bộ.

12.2 Ứng dụng kiểu client-server

Khái niệm:

* Ứng dụng được chia thành hai phần:
  + Client (máy khách): gửi yêu cầu, hiển thị giao diện.
  + Server (máy chủ): xử lý yêu cầu, trả về kết quả.
* Giao tiếp qua mạng (HTTP, TCP/IP).

Đặc điểm:

* Phân vai trò rõ ràng: client lo UI, server lo xử lý.
* Yêu cầu kết nối mạng: để client giao tiếp với server.

Ví dụ:

* Ứng dụng web, email, trò chơi online, hệ thống quản trị cơ sở dữ liệu.

12.3 Ứng dụng web dưới góc nhìn lập trình viên

Cấu trúc ứng dụng web:

* Là ứng dụng client-server:
  + Client: trình duyệt web (Chrome, Firefox, Edge...).
  + Server: web server (như Express, Apache...).

Vai trò lập trình viên:

* Front-end (client-side):
  + Dùng HTML, CSS, JavaScript.
  + Làm giao diện, hiệu ứng, xử lý sự kiện người dùng.
  + Gọi là front-end developer.
* Back-end (server-side):
  + Dùng ngôn ngữ như JavaScript (Node.js), Python, PHP, Java...
  + Xử lý logic, thao tác cơ sở dữ liệu.
  + Gọi là back-end developer.
* Full-stack developer:
  + Làm được cả front-end và back-end.

Lưu ý:

* Dù làm ở mảng nào, lập trình viên cũng nên hiểu căn bản về phần còn lại để phối hợp hiệu quả.

Cấu trúc trình duyệt web:

1. [1] Thanh menu và công cụ trình duyệt.
2. [2] Thanh địa chỉ: nơi nhập URL, gửi request.
3. [3] Vùng hiển thị giao diện web/ứng dụng.

***BT Web back-end (12) - Ứng dụng client-server***

Câu 12.1: Ứng dụng cục bộ là gì? Phát biểu nào sau đây không đúng?

Ứng dụng cục bộ (local application) là ứng dụng chạy trực tiếp trên thiết bị người dùng, không phụ thuộc vào mạng hay server bên ngoài.

A. Ứng dụng cục bộ chạy trực tiếp trên thiết bị của người dùng mà không cần kết nối liên tục với máy chủ ở xa.  
Đúng – Đây là đặc điểm chính của ứng dụng cục bộ.

B. Ứng dụng cục bộ luôn yêu cầu kết nối internet ổn định để hoạt động hiệu quả.  
Sai – Không đúng! Ứng dụng cục bộ không cần internet để hoạt động. Nó có thể chạy hoàn toàn offline, ví dụ như Microsoft Word, phần mềm kế toán, game offline, v.v.

C. Ứng dụng cục bộ có thể truy cập trực tiếp vào các tài nguyên của thiết bị như hệ thống tập tin và phần cứng.  
Đúng – Ứng dụng cục bộ có thể đọc ghi tệp, dùng camera, GPS...

D. Dữ liệu của ứng dụng cục bộ thường được lưu trữ trên thiết bị của người dùng, giúp tăng cường bảo mật và quyền riêng tư.  
Đúng – Vì không lưu trữ trên server, dữ liệu nằm trong máy người dùng nên bảo mật cao hơn (nếu người dùng kiểm soát thiết bị).

Đáp án: B – vì phát biểu này không đúng với ứng dụng cục bộ.

Câu 12.2: Ứng dụng kiểu client-server là gì? Phát biểu nào sau đây không đúng?

Ứng dụng client-server là kiểu ứng dụng phân chia vai trò giữa client và server, trong đó client gửi yêu cầu, và server xử lý, phản hồi.

A. Client tập trung chủ yếu vào việc hiển thị giao diện người dùng, trong khi server tập trung xử lý logic và dữ liệu.  
Đúng – Đây là đặc trưng của mô hình client-server.

B. Giao tiếp giữa client và server thường được thực hiện qua mạng, sử dụng các giao thức như HTTP, WebSocket hoặc TCP/IP.  
Đúng – Những giao thức như HTTP là nền tảng giao tiếp client-server.

C. Client và server luôn phải chạy trên các thiết bị riêng biệt, không bao giờ có thể cùng chạy trên một máy tính.  
Sai – Client và server *có thể* chạy cùng một máy, ví dụ: khi bạn lập trình backend bằng Node.js trên localhost, client và server đều ở trên máy của bạn.

D. Server có nhiệm vụ nhận yêu cầu từ client, xử lý chúng, và gửi phản hồi lại cho client.  
Đúng – Đây là vai trò cơ bản của server trong mô hình này.

Đáp án: C – vì phát biểu này không đúng với thực tế về client-server.

**Web back-end (13) - Lập trình client-server**

13. Lập trình client-server

13.1 Gửi request tới web server

* Mục tiêu: Tìm hiểu cơ chế giao tiếp giữa client và server trong ứng dụng web.

Các bước xử lý cơ bản trên web server:

1. Nhận request từ client
2. Phân tích request, chuyển đến ứng dụng xử lý
3. Đọc dữ liệu từ cơ sở dữ liệu
4. Đổ dữ liệu vào HTML template
5. Gửi response (HTML chứa dữ liệu) về client
6. Trình duyệt hiển thị nội dung
7. Trình duyệt tự gửi các request khác để lấy nội dung tĩnh (CSS, JS, hình ảnh,...)

Lập trình từng bước:

Bước 1: Khởi động web server

* Dùng Express (Node.js)
* Mã nguồn index.js:

'use strict'

const express = require('express');

const app = express();

const port = process.env.PORT || 9000;

app.listen(port, () => {

console.log(`server dang chay tren cong ${port}`);

});

* Chạy bằng lệnh node index.js hoặc nodemon

Bước 2: Gửi request từ trình duyệt

* Truy cập: http://localhost:9000/?name=shoes&size=40
* Nếu chưa có xử lý, trình duyệt báo: Cannot GET /

Giải thích URL:

* Giao thức: http
* Domain: localhost
* Cổng: 9000
* Path: /
* Query string: ?name=shoes&size=40

13.2 Gửi response về client

Bước 3, 4, 5: Nhận request, phân tích và xử lý

* Dùng phương thức app.get() của Express để xử lý request kiểu GET:

app.get("/", (req, res) => {

const product = req.query.name;

const size = req.query.size;

res.send(`Bạn muốn mua ${product} cỡ ${size}`);

});

Bước 6: Trình duyệt hiển thị nội dung

* Khi truy cập: http://localhost:9000/?name=shoes&size=40
* Trình duyệt sẽ hiển thị: Bạn muốn mua shoes cỡ 40

Ghi nhớ:

* app.listen() luôn chạy sau cùng nhưng sẽ được thực thi đầu tiên khi server khởi động.
* Các hàm như app.get() dùng để xử lý logic khi có request tới.

***BT Web back-end (13) - Lập trình client-server***

Câu 13.1: Viết đoạn mã xử lý GET request có MSSV, HoTen, DiemTrungBinh

Dưới đây là đoạn mã Node.js + Express dùng để xử lý yêu cầu từ trình duyệt:

Bước 1: Cài đặt Express (nếu chưa có)

npm init -y

npm install express

Bước 2: Tạo file server.js và viết đoạn mã sau:

const express = require('express');

const app = express();

// Định nghĩa cổng

const PORT = 3000;

// Xử lý GET request với tham số truyền qua URL (query string)

app.get('/', (req, res) => {

const mssv = req.query.MSSV;

const hoTen = req.query.HoTen;

const diemTrungBinh = req.query.DiemTrungBinh;

// Xử lý: chuyển họ tên có dấu sang không dấu nếu cần, ở đây ví dụ thay "Nguyễn Văn Tèo" thành "Nguyen Van Teo"

const tenKhongDau = hoTen.normalize("NFD").replace(/[\u0300-\u036f]/g, "");

const diem = parseFloat(diemTrungBinh) \* 10;

const message = `

Chào bạn ${tenKhongDau}

MSSV: ${mssv}

Điểm trung bình của bạn là: ${diem}

Chúc bạn học chăm!

`;

res.send(`<pre>${message}</pre>`);

});

app.listen(PORT, () => {

console.log(`Server đang chạy tại http://localhost:${PORT}`);

});

Cách chạy:

Mở trình duyệt và truy cập:

http://localhost:3000?MSSV=11223344&HoTen=Nguyễn Văn Tèo&DiemTrungBinh=7.0

Câu 13.2: URL là gì? Phát biểu nào sau đây không đúng?

Phân tích từng phát biểu:

A. URL là một địa chỉ duy nhất xác định vị trí của một tài nguyên trên Internet.  
Đúng.- Đây là định nghĩa cơ bản và chính xác về URL. Ví dụ: https://www.google.com là một URL xác định trang Google.

B. URL bao gồm các thành phần như giao thức, tên miền, đường dẫn, truy vấn và phân mảnh.  
Đúng.- Ví dụ URL: https://example.com/path/page.html?search=abc#top

C. Phần "truy vấn" (query) trong URL được sử dụng để truyền dữ liệu đến máy chủ thông qua các tham số.  
Đúng.- Trong ví dụ ?MSSV=11223344&HoTen=Teo, phần này được dùng để gửi dữ liệu cho server.

D. Giao thức "FTP" là giao thức phổ biến nhất được sử dụng trong URL để truy cập các trang web.  
Sai.- Giao thức phổ biến nhất là HTTP hoặc HTTPS, không phải FTP. FTP chỉ được dùng để truyền tải file.

Đáp án là: D

**Web back-end (14) - Lập trình hướng sự kiện**

14. Lập trình hướng sự kiện

14.1 Khái niệm lập trình hướng sự kiện (Event-driven programming)

* Là mô hình lập trình mà chương trình không chạy tuần tự từ trên xuống dưới, mà đợi các sự kiện xảy ra để thực thi các hàm xử lý tương ứng.
* Sự kiện có thể đến từ người dùng, hệ thống, hoặc các chương trình khác.

14.2 Trong Node.js

* Node.js thiết kế dựa trên mô hình hướng sự kiện, đặc biệt hiệu quả với các ứng dụng I/O-intensive (đọc/ghi file, request mạng...).
* Cấu trúc gồm:
  + Event: hành động/sự thay đổi (như có request đến).
  + EventEmitter: bộ quản lý sự kiện, cho phép phát và lắng nghe sự kiện.
  + Event Loop: vòng lặp sự kiện, kiểm tra và thực thi các hàm callback khi có sự kiện.
  + Event Handler: hàm xử lý khi sự kiện xảy ra.

Phương thức trong EventEmitter

* on(eventName, listener): đăng ký lắng nghe một sự kiện.
* emit(eventName, [args]): phát sinh một sự kiện.

14.3 Thực hành

Ví dụ 1: Hàm thường

const events = require('events');

const eventEmitter = new events.EventEmitter();

eventEmitter.on('hetGio', xuLyThongBao);

function xuLyThongBao(tb) {

console.log(tb);

}

setTimeout(() => {

eventEmitter.emit('hetGio', 'Hết giờ học rồi, về thôi!!!!!!!');

}, 2000);

Ví dụ 2: Dùng arrow function

const events = require('events');

const eventEmitter = new events.EventEmitter();

eventEmitter.on('hetGio', (tb) => {

console.log(tb);

});

setTimeout(() => {

eventEmitter.emit('hetGio', 'Hết giờ học rồi, về thôi!!!!!!!');

}, 2000);

Kết quả: sau 2 giây, console in ra thông báo "Hết giờ học rồi, về thôi!!!!!!!"

***BT Web back-end (14) - Lập trình hướng sự kiện***

Câu 14.2

Cách 1: Sử dụng hàm thông thường

const EventEmitter = require('events'); // Nhập module events

// Khai báo đối tượng EventEmitter

const eventEmitter = new EventEmitter();

// Khai báo sự kiện vaoLop với hàm xử lý xuLyVaoLop

function xuLyVaoLop() {

console.log("Đã đến giờ học");

}

// Đăng ký sự kiện vaoLop

eventEmitter.on('vaoLop', xuLyVaoLop);

// Sử dụng setTimeout để gọi sự kiện vaoLop sau 5 giây

setTimeout(() => {

eventEmitter.emit('vaoLop'); // Phát ra sự kiện vaoLop

}, 5000); // 5 giây

Cách 2: Sử dụng hàm mũi tên

const EventEmitter = require('events'); // Nhập module events

// Khai báo đối tượng EventEmitter

const eventEmitter = new EventEmitter();

// Khai báo sự kiện vaoLop với hàm xử lý bằng hàm mũi tên

eventEmitter.on('vaoLop', () => {

console.log("Đã đến giờ học");

});

// Sử dụng setTimeout để gọi sự kiện vaoLop sau 5 giây

setTimeout(() => {

eventEmitter.emit('vaoLop'); // Phát ra sự kiện vaoLop

}, 5000); // 5 giây

Giải thích:

1. EventEmitter: Lớp EventEmitter giúp phát và lắng nghe sự kiện.
2. setTimeout(): Được dùng để trì hoãn việc phát ra sự kiện vaoLop sau 5 giây.
3. emit(): Phát ra sự kiện và gọi hàm xử lý tương ứng.

Câu 14.3: Mô hình lập trình hướng sự kiện trong Node.js là gì? Phát biểu nào sau đây không đúng?

A. Event Emitter liên tục kiểm tra hàng đợi sự kiện và thực thi các hàm callback khi có sự kiện xảy ra.

Sai. Vì:

* Event Emitter chỉ là một lớp/đối tượng dùng để phát và lắng nghe sự kiện chứ không có nhiệm vụ kiểm tra hàng đợi sự kiện.
* Việc kiểm tra hàng đợi sự kiện là nhiệm vụ của Event Loop, không phải Event Emitter.

B. Event là các hành động hoặc sự thay đổi trạng thái xảy ra trong ứng dụng.

Đúng. Ví dụ: người dùng nhấn nút, server nhận request, đọc xong file, v.v…

C. Event Loop liên tục kiểm tra hàng đợi sự kiện và thực thi các hàm callback khi có sự kiện xảy ra.

Đúng. Đây là trung tâm xử lý bất đồng bộ trong Node.js – nó giúp Node.js không bị chặn khi đang chờ một tác vụ hoàn thành.

D. Event Handler là các hàm được gọi khi một sự kiện xảy ra.

Đúng. Ví dụ: khi sự kiện vaoLop xảy ra thì hàm xuLyVaoLop() được gọi.

Đáp án: A (vì đây là phát biểu không đúng)

**Web back-end (15) - Event-driven, Route handler và Middleware trong Express**

15. Event-driven, Route handler và Middleware trong Express"

15.1 Lập trình hướng sự kiện trong Express

* Express xây dựng trên nền tảng Node.js và sử dụng mô hình lập trình hướng sự kiện.
* Mỗi request (GET, POST, PUT, DELETE) từ client được xem là một sự kiện.
* Các module như http, fs, stream là các Event Emitter (bộ quản lý sự kiện) của Express, giúp phát ra các sự kiện.
* Vòng lặp sự kiện (Event Loop) trong Node.js giúp xử lý các sự kiện một cách bất đồng bộ (asynchronous), không chặn luồng (non-blocking).
* Route handler và middleware đóng vai trò như Event Handler, thực thi các hàm khi các sự kiện xảy ra.

15.2 Thực hành với GET request

1. GET Request xử lý:
   * Khi client gửi GET request đến server, Express sẽ kiểm tra URL và method (GET) để xác định route và gọi hàm callback (route handler) phù hợp.
   * Route handler nhận request và trả về response cho client.

Ví dụ:

app.get('/', (req, res) => {

const name = req.query.name;

res.send(`Web server chào bạn ${name}`);

});

* + Khi truy cập URL http://localhost:9000/?name=Teo, server trả về "Web server chào bạn Teo".

1. Middleware cho GET request:
   * Middleware là các hàm trung gian, thực thi trước hoặc sau route handler. Trong ví dụ này, middleware ghi log mỗi khi có GET request.
   * Middleware có thể chuyển tiếp request đến route handler tiếp theo bằng next().

Ví dụ Middleware:

app.use((req, res, next) => {

console.log(`[${new Date().toISOString()}] nhận GET request tại ${req.url}`);

next(); // Chuyển tiếp request đến route handler

});

* + Kết quả trên console khi truy cập URL sẽ ghi nhận thời gian và đường dẫn: [2025-03-31T11:54:20.737Z] nhận GET request tại /?name=Teo.

15.3 Route handler và Middleware trong Express

1. Route Handler:
   * Route handler là các hàm xử lý cho các HTTP request tại một route cụ thể (GET, POST, PUT, DELETE).
   * Nhận 3 tham số: req (request), res (response), và next (chuyển tiếp request tới middleware hoặc handler tiếp theo).
   * Cú pháp: app.METHOD(path, callback);
   * Route handler xử lý các logic nghiệp vụ như truy vấn cơ sở dữ liệu hoặc trả về kết quả cho client.

Ví dụ Route handler:

app.get('/', (req, res) => {

const name = req.query.name;

res.send(`Web server chào bạn ${name}`);

});

* + Route handler nối tiếp: Có thể nối nhiều hàm xử lý bằng next().

Ví dụ nối tiếp Route handler:

app.get('/profile', (req, res, next) => {

console.log('Kiểm tra quyền truy cập...');

req.user = { name: "Ti" };

next(); // Chuyển tiếp

}, (req, res) => {

res.send(`Chào bạn: ${req.user.name}`);

});

* + Kết quả truy cập URL /profile: "Chào bạn: Ti".

1. Middleware:
   * Middleware là các hàm thực thi trong quá trình xử lý một HTTP request, có thể ở trước hoặc sau route handler.
   * Chức năng:
     + Xử lý request (xác thực, ghi log).
     + Sửa đổi đối tượng req hoặc res.
     + Dừng luồng xử lý hoặc chuyển tiếp request bằng next().
   * Middleware giúp tách biệt các logic chung khỏi route handler.

Ví dụ Middleware:

app.use((req, res, next) => {

console.log(`[${new Date().toISOString()}] nhận GET request tại ${req.url}`);

next(); // Chuyển tiếp đến route handler

});

Các loại Middleware:

* + Application-level middleware: Áp dụng cho toàn bộ ứng dụng với app.use().
  + Router-level middleware: Áp dụng cho nhóm route cụ thể bằng express.Router().
  + Error-handling middleware: Xử lý lỗi, nhận tham số err.
  + Built-in middleware: Các middleware do Express cung cấp như express.json(), express.static().
  + Third-party middleware: Các middleware của bên thứ ba như body-parser, cors, morgan.

Ví dụ middleware mức ứng dụng:

app.use((req, res, next) => {

req.timestamp = new Date();

next(); // Chuyển tiếp đến route handler

});

app.get('/', (req, res) => {

res.send(`Khách truy cập web lúc ${req.timestamp.toISOString()}`);

});

***BT Web back-end (15) - Event-driven, Route handler và Middleware trong Express***

Câu hỏi 15.2 - Mô hình lập trình hướng sự kiện của Express. Phát biểu nào sau đây không đúng?

A: Đúng: Mỗi request (GET, POST, PUT, DELETE) từ client đều được coi là một sự kiện (event) mà Express lắng nghe và xử lý.

B: Đúng: Các module như http, fs, stream là những Event Emitter trong Node.js. Express được xây dựng trên nền Node.js nên tận dụng được chúng.

C: Sai: Express sử dụng Vòng lặp sự kiện (Event Loop) của Node.js, không phải của trình duyệt. Trình duyệt có event loop riêng nhưng không liên quan đến môi trường máy chủ của Express.

D: Đúng: Các route handler và middleware được gọi khi sự kiện tương ứng xảy ra, giống như các hàm xử lý sự kiện.

Đáp án: C.

Câu hỏi 15.3 - Route handler trong Express là gì? Phát biểu nào sau đây không đúng?

A. Đúng: Route handler chỉ được gọi khi request khớp với route và phương thức HTTP (GET, POST,...) đã định nghĩa.

B. Đúng: Route handler thường thực hiện các logic như truy vấn CSDL, xử lý dữ liệu,...

C. Đúng: Có thể gọi next() để chuyển sang middleware/handler tiếp theo.

D. Sai: Một route có thể có nhiều route handler (chuỗi các middleware/handler). Ví dụ:

app.get('/user', middleware1, middleware2, handler);

Đáp án: D.

Câu hỏi 15.4 - Middleware trong Express là gì? Phát biểu nào sau đây không đúng?

A. Đúng: Middleware được thực thi trước hoặc sau khi request đến route handler.

B. Đúng: Middleware có thể xử lý việc xác thực, ghi log, parse dữ liệu,...

C. Đúng: Middleware có thể sửa đổi req, res.

D. Sai: Middleware thường nhận 3 hoặc 4 tham số, không chỉ req, res:

* + Bình thường: (req, res, next)
  + Middleware xử lý lỗi: (err, req, res, next)

Đáp án đúng: D.