BỘ CÂU HỎI MODULE-1

|  |  |  |
| --- | --- | --- |
| STT | CÂU HỎI | ĐÁP ÁN |
| 1 | Các kiến thức ở module 1 | * Git * Flowchart & mã giả * JS * HTML (Hyper Text Markup Language) * Biến, kiểu dữ liệu, toán tử * Cấu trúc điều kiện, lặp * Mảng, hàm * Lặp trình hướng đối tượng * Thuật toán |
| 2 | Giải thuật/Thuật toán là gì? | Là một tập hợp các bước, quy tắc hoặc hướng dẫn được xác định rõ ràng và sắp xếp theo thứ tự để giải quyết một vấn đề cụ thể hoặc thực hiện một nhiệm vụ nào đó |
| 3 | So sánh việc trình bày giải thuật bằng pseudo-code và flowchart? | | **Tiêu chí** | **Pseudo-code** | **Flowchart** | | --- | --- | --- |  |  |  |  | | --- | --- | --- | | **Tính dễ hiểu** | - Dễ hiểu với người có kiến thức lập trình cơ bản. | - Trực quan, dễ hiểu với cả người không biết lập trình. |  |  |  |  | | --- | --- | --- | | **Mức độ trực quan** | - Không trực quan, tập trung vào logic và trình tự. | - Rất trực quan nhờ các hình khối và mũi tên. |  |  |  |  | | --- | --- | --- | | **Chi tiết** | - Có thể trình bày chi tiết từng bước logic. | - Phù hợp để biểu diễn logic đơn giản. Các bước phức tạp dễ làm sơ đồ rối. |  |  |  |  | | --- | --- | --- | | **Khả năng chuyển đổi sang code** | - Dễ chuyển đổi thành mã lập trình thực tế. | - Chuyển đổi sang code phức tạp hơn, cần diễn giải từ sơ đồ. |  |  |  |  | | --- | --- | --- | | **Ứng dụng** | - Phù hợp cho lập trình viên và nhà phát triển hệ thống. | - Phù hợp để giải thích ý tưởng cho người không chuyên. |  |  |  |  | | --- | --- | --- | | **Thời gian xây dựng** | - Nhanh, dễ sửa đổi và cập nhật. | - Tốn thời gian để thiết kế và chỉnh sửa sơ đồ. |  |  |  |  | | --- | --- | --- | | **Khả năng biểu diễn vòng lặp/nhánh rẽ** | - Gọn gàng, dễ biểu diễn logic phức tạp. | - Cần nhiều khối để biểu diễn các vòng lặp và nhánh, dễ gây rối. | |
| 4 | Tại sao lại cần trình bày giải thuật trước khi bắt tay vào code? | Trình bày giải thuật trước khi lập trình không chỉ giúp **hiểu vấn đề**, mà còn giúp **tối ưu hóa quy trình**, **giảm thiểu sai sót**, và **tăng hiệu quả làm việc**. Đây là bước cần thiết để tạo ra một phần mềm chất lượng cao |
| 5 | Tại sao lại nên sử dụng flowchart hơn pseudo-code khi thể hiện giải thuật của bài toán phức tạp? | * Flowchart thích hợp cho các bài toán phức tạp vì:   + Trực quan và dễ hiểu.  + Phù hợp để phân tích, thảo luận và trình bày ý tưởng.  + Hỗ trợ quản lý và làm giảm độ phức tạp của giải thuật.   * Pseudo-code tốt hơn khi   + Tập trung vào lập trình hoặc triển khai.  + Logic giải thuật không quá phức tạp. |
| 6 | Tại sao giải thuật lại cần hữu hạn các bước thực thi? | 1.Đảm bảo kết thúc và trả về kết quả.  2.Đảm bảo khả thi trong thực tế.  3.Tuân theo định nghĩa và mục tiêu của giải thuật.  Giải thuật vô hạn là không chấp nhận được vì nó không thể áp dụng vào thực tiễn và không đáp ứng yêu cầu bài toán. |
| 7 | Trình bày ý nghĩa các ký hiệu sử dụng khi vẽ flowchart? | · Elip: Bắt đầu và kết thúc.  · Bình hành: Nhập và in dữ liệu.  · Hình thoi: Kiểm tra điều kiện .  · Mũi tên: Thể hiện luồng logic.  · Hình chữ nhật: Thể hiện một bước thực thi hoặc hoạt động (Process).  · Hình tròn nhỏ (Connector): Kết nối các phần của flowchart khi quy trình quá dài hoặc phức tạp. |
| 8 | Vẽ giải thuật giải phương trình bậc nhất. | Ax + B = 0 |
| 9 | Bạn biết gì về VCS – Version Control System (hệ thống quản lý phiên bản)? | VSC là hệ thống hỗ trợ:   * Khôi phục lại phiên bản cũ của các file thạm chí là toàn bộ dự án * Xem lại các thay đổi theo thời gian. * Xem lại người thực hiện cuối để có thể xác định nơi gây sự cố * Khôi phục lại file nếu vô tình xóa |
| 10 | Phân biệt hệ thống quản lý mã nguồn tập trung và hệ thống quản lý mã nguồn phân tán? | Hệ thống quản lý mã nguồn tập trung (Centralized Version Control System - CVCS) và hệ thống quản lý mã nguồn phân tán (Distributed Version Control System - DVCS) đều là các công cụ hỗ trợ quản lý phiên bản mã nguồn trong dự án. Tuy nhiên, chúng khác nhau về cấu trúc và cách thức hoạt động.  1. Hệ thống quản lý mã nguồn tập trung (CVCS)  Đặc điểm:  Kiến trúc tập trung: Có một máy chủ trung tâm chứa toàn bộ kho mã nguồn và các lịch sử thay đổi.  Cách hoạt động: Các máy khách kết nối với máy chủ để truy cập mã nguồn, tải xuống hoặc tải lên các thay đổi.  Ví dụ: Subversion (SVN), Perforce, CVS.  Ưu điểm:   1. Dễ triển khai và quản lý: Máy chủ trung tâm duy nhất, dễ kiểm soát.   2. Thích hợp cho dự án nhỏ: Dễ sử dụng cho nhóm nhỏ với ít thay đổi đồng thời.  3. Tập trung hóa dữ liệu: Tất cả thay đổi và mã nguồn được lưu trữ tại một nơi duy nhất.  Nhược điểm:   1. Phụ thuộc vào máy chủ: Nếu máy chủ bị lỗi hoặc mất kết nối, các nhà phát triển không thể truy cập mã nguồn hoặc đẩy thay đổi.   2. Hiệu suất kém với đội ngũ lớn: Dễ bị tắc nghẽn khi có nhiều người cùng làm việc.  3. Không hỗ trợ làm việc ngoại tuyến tốt: Mọi thao tác đều yêu cầu kết nối với máy chủ.  4. Rủi ro bảo mật: Mọi dữ liệu được lưu trên một máy chủ duy nhất, dễ trở thành mục tiêu tấn công.  2. Hệ thống quản lý mã nguồn phân tán (DVCS)  Đặc điểm:  Kiến trúc phân tán: Mỗi máy khách giữ một bản sao đầy đủ của kho mã nguồn, bao gồm lịch sử thay đổi.  Cách hoạt động: Nhà phát triển có thể làm việc ngoại tuyến trên bản sao cục bộ và chỉ cần kết nối với máy chủ khi đồng bộ thay đổi.  Ví dụ: Git, Mercurial, Bazaar.  Ưu điểm:   1. Hỗ trợ làm việc ngoại tuyến: Người dùng có thể commit, xem lịch sử, và thực hiện các thay đổi cục bộ mà không cần kết nối internet. 2. Hiệu suất tốt hơn: Các thao tác (như commit, revert) thực hiện nhanh chóng vì diễn ra trên máy cục bộ. 3. Phân tán rủi ro: Mỗi máy khách đều lưu trữ toàn bộ lịch sử kho mã, giảm rủi ro mất dữ liệu nếu máy chủ bị lỗi. 4. Hỗ trợ làm việc nhóm tốt hơn: Hỗ trợ dễ dàng hợp nhất mã (merge) và giải quyết xung đột (conflict). 5. Phù hợp với dự án lớn: Xử lý tốt các nhóm phát triển lớn, có nhiều nhánh (branch) và phiên bản.   Nhược điểm:   1. Phức tạp hơn: Cần nhiều thao tác (pull, push, fetch, merge, ...) để đồng bộ giữa máy cục bộ và máy chủ. 2. Chiếm nhiều không gian lưu trữ: Mỗi máy chứa toàn bộ lịch sử kho mã nguồn, tốn dung lượng hơn so với CVCS. 3. Cần đào tạo: Đòi hỏi người dùng hiểu rõ quy trình làm việc với hệ thống phân tán.   So sánh CVCS và DVCS   |  |  |  | | --- | --- | --- | | Tiêu chí | CVCS | DVCS | | Kiến trúc | Tập trung | Phân tán | | Máy chủ trung tâm | Bắt buộc (yếu tố quyết định) | Không bắt buộc (chỉ để đồng bộ) | | Làm việc ngoại tuyến | Không hỗ trợ | Hỗ trợ đầy đủ | | Sao lưu dữ liệu | Chỉ có trên máy chủ trung tâm | Có trên mỗi máy khách | | Hiệu suất | Chậm với dự án lớn | Tốt hơn với nhóm lớn | | Merge và Branching | Ít linh hoạt | Mạnh mẽ và linh hoạt | | Phức tạp | Dễ sử dụng hơn | Khó học và quản lý hơn ban đầu |     Khi nào sử dụng CVCS?  + Dự án nhỏ, ít thành viên.  + Yêu cầu kiểm soát chặt chẽ và dễ dàng quản lý.  + Môi trường làm việc chủ yếu là trực tuyến.  Khi nào sử dụng DVCS?  + Dự án lớn, nhiều nhánh và thay đổi đồng thời.  + Yêu cầu làm việc ngoại tuyến hoặc làm việc nhóm ở các khu vực địa lý khác nhau.  + Cần tính linh hoạt cao, dễ merge mã và giải quyết xung đột. |
| 11 | Git là gì? Tại sao cần sử dụng git? | Git là một hệ thống quản lý phiên bản phân tán (Distributed Version Control System - DVCS) được sử dụng để theo dõi các thay đổi trong mã nguồn của dự án và phối hợp làm việc giữa nhiều lập trình viên.  Các tính năng nổi bật của Git  1. Theo dõi thay đổi: Lưu trữ lịch sử các phiên bản của mã nguồn, giúp bạn xem lại, khôi phục, hoặc so sánh các thay đổi.  2. Làm việc phân tán: Cho phép nhiều người làm việc trên cùng một dự án mà không cần kết nối liên tục với máy chủ.  3. Tốc độ cao: Các thao tác như commit, merge, hay checkout được thực hiện nhanh chóng.  4. Branching mạnh mẽ: Dễ dàng tạo và quản lý các nhánh (branch) để phát triển tính năng mới mà không ảnh hưởng đến mã nguồn chính.  5. Hỗ trợ nhiều nền tảng: Hoạt động tốt trên Windows, macOS và Linux.  ---  Tại sao cần sử dụng Git?  1. Quản lý mã nguồn hiệu quả:  - Git giúp bạn tổ chức và lưu trữ các phiên bản mã nguồn, tránh mất mát dữ liệu hoặc ghi đè mã khi nhiều người cùng làm việc.  - Có thể quay lại phiên bản trước nếu cần.  2. Làm việc nhóm dễ dàng:  - Cho phép nhiều người cùng làm việc trên một dự án mà không gặp xung đột mã.  - Dễ dàng đồng bộ các thay đổi thông qua các hệ thống lưu trữ như GitHub, GitLab, hoặc Bitbucket.  3. Hỗ trợ phát triển nhanh chóng:  - Với tính năng branching và merging, bạn có thể phát triển và thử nghiệm các tính năng mới mà không lo ảnh hưởng đến mã nguồn chính.  - Các xung đột được giải quyết dễ dàng thông qua hệ thống thông báo và kiểm tra.  4. Bảo mật cao:  - Lịch sử commit của Git là bất biến (không thể thay đổi) và được mã hóa, giúp bạn bảo vệ được lịch sử làm việc.  5. Tiêu chuẩn trong ngành phần mềm:  - Git đã trở thành công cụ tiêu chuẩn trong hầu hết các dự án, nên việc sử dụng Git giúp bạn thích nghi nhanh với các quy trình làm việc chuyên nghiệp. |
| 12 | Git thuộc hệ thống quản lý mã nguồn tập trung hay phân tán? | Git thuộc hệ thống quản lý mã nguồn phân tán |
| 13 | Repository là gì? Phân biệt LocalRepository và RemoteRepository? | Repository (repo) là nơi lưu trữ toàn bộ mã nguồn và lịch sử thay đổi của một dự án trong Git. Nó bao gồm các file dự án và dữ liệu quản lý phiên bản mà Git sử dụng để theo dõi các thay đổi, thường được lưu trong một thư mục ẩn `.git`.   |  |  |  | | --- | --- | --- | | Tiêu chí | Local Repo | Remote Repo | | Định nghĩa | Repository nằm trên máy tính cục bộ của bạn (local machine). | Repository được lưu trên một máy chủ từ xa (remote server), thường sử dụng các dịch vụ như GitHub, GitLab,... | | Vị trí lưu trữ | Lưu trữ trên máy cá nhân của lập trình viên, trong thư mục làm việc. | Lưu trữ trên máy chủ hoặc trên nền tảng trực tuyến (cloud). | | Mục đích sử dụng | Để chỉnh sửa, phát triển và quản lý phiên bản cục bộ. | Để chia sẻ mã nguồn và phối hợp làm việc nhóm từ nhiều máy khác nhau. | | Khả năng truy cập | Chỉ có thể truy cập trên máy tính nơi lưu repository. | Có thể truy cập từ bất kỳ đâu qua internet. | |
| 14 | Đồng bộ giữa các repository (local và remote) để làm gì? | 1. Chia sẻ mã nguồn và cập nhật công việc 2. Hợp tác làm việc nhóm 3. Bảo vệ dữ liệu 4. Quản lý phiên bản hiệu quả 5. Phát triển đồng thời trên nhiều máy |
| 15 | Thao tác push của git để làm gì? Cú pháp câu lệnh? | Gửi các thay đổi từ Local Repository lên Remote Repository (git push) |
| 16 | Thao tác pull của git để làm gì? | Lấy thay đổi từ Remote Repository về Local Repository (git pull). |
| 17 | Thao tác clone của git để làm gì? Cú pháp câu lệnh? | Tải repository từ remote về máy cục bộ (git clone). |
| 18 | Nêu tên và ý nghĩa các câu lệnh cơ bản đã được học để làm việc với git? | git clone [URL của Remote Repository]: tạo bản sao được liên kết với repo  git pull: cập nhật mã nguồn từ một Remote Repository về Local Repository  git push: đẩy các commit từ local về central  git checkout <id commit>: di chuyển giữa các commit  git branch <tên nhánh>: tạo nhánh  git checkout <tên nhánh>: di chuyển giữa các nhánh |
| 19 | www là gì? | (World Wide Web) là một hệ thống thông tin toàn cầu, cho phép người dùng truy cập và chia sẻ tài nguyên như văn bản, hình ảnh, video, và các dữ liệu khác thông qua internet.  Các thành phần chính của WWW:  1. Website:  - Là tập hợp các trang web (web pages) được liên kết với nhau và được truy cập qua một địa chỉ duy nhất (URL).  - Ví dụ: Google (www.google.com) hay Wikipedia (www.wikipedia.org).  2. HTTP/HTTPS (HyperText Transfer Protocol):  - Giao thức truyền thông để chuyển dữ liệu giữa trình duyệt và máy chủ.  - HTTP: Không mã hóa dữ liệu.  - HTTPS: Bảo mật hơn nhờ sử dụng mã hóa.  3. HTML (HyperText Markup Language):  - Ngôn ngữ đánh dấu để xây dựng cấu trúc và nội dung của các trang web.  4. URL (Uniform Resource Locator):  - Địa chỉ duy nhất của mỗi trang web trên WWW.  5. Trình duyệt web (Web Browser):  - Phần mềm giúp người dùng truy cập các trang web, ví dụ: Google Chrome, Firefox, hoặc Safari.  ---  WWW hoạt động như thế nào?  1. Người dùng nhập URL vào thanh địa chỉ của trình duyệt (ví dụ: `www.example.com`).  2. Trình duyệt gửi yêu cầu (request) tới máy chủ thông qua giao thức HTTP hoặc HTTPS.  3. Máy chủ xử lý yêu cầu và trả về nội dung (dữ liệu HTML, CSS, JavaScript,...).  4. Trình duyệt hiển thị nội dung trang web cho người dùng.  ---  WWW khác gì so với Internet?  - Internet: Là mạng lưới toàn cầu kết nối các thiết bị với nhau. Nó là cơ sở hạ tầng vật lý cho các dịch vụ như email, truyền file, và cả WWW.  - WWW: Là một trong những dịch vụ hoạt động trên nền Internet, tập trung vào truy cập và chia sẻ thông tin qua các trang web.  ---  Tại sao WWW quan trọng?  1. Dễ dàng truy cập thông tin:  - Cung cấp thông tin từ khắp nơi trên thế giới chỉ với vài cú nhấp chuột.  2. Hỗ trợ giao tiếp:  - WWW cho phép giao tiếp qua các nền tảng như mạng xã hội, blog, và diễn đàn.  3. Thúc đẩy thương mại và giáo dục:  - Các dịch vụ thương mại điện tử, học trực tuyến, và chia sẻ tri thức đều dựa vào WWW.  4. Kết nối toàn cầu:  - Tạo điều kiện cho sự kết nối giữa các quốc gia, văn hóa và cộng đồng. |
| 20 | HTML là gì? | HTML (HyperText Markup Language) là ngôn ngữ đánh dấu siêu văn bản được sử dụng để tạo và cấu trúc các thành phần trên trang web hoặc ứng dụng. HTML không phải là ngôn ngữ lập trình mà là ngôn ngữ đánh dấu, giúp phân chia các đoạn văn, tiêu đề, liên kết, hình ảnh, và các phần tử khác trên trang web |
| 21 | Nêu tên các thẻ làm việc với văn bản trong tài liệu html? | Trong HTML, có nhiều thẻ được sử dụng để định dạng và làm việc với văn bản. Dưới đây là một số thẻ phổ biến:  **Thẻ tiêu đề (Heading Tags)**: <h1> đến <h6>: Định dạng các tiêu đề từ lớn nhất (<h1>) đến nhỏ nhất (<h6>).  **Thẻ đoạn văn (Paragraph Tag)**: <p>: Định dạng một đoạn văn bản.  **Thẻ ngắt dòng (Line Break Tag)**: <br>  **Thẻ đường ngang (Horizontal Rule Tag)**: <hr>: Chèn một đường ngang để phân chia nội dung.  **Thẻ in đậm (Bold Tag)**:   1. <b>: In đậm văn bản. 2. <strong>: In đậm và nhấn mạnh văn bản, có ý nghĩa quan trọng hơn trong SEO.   **Thẻ in nghiêng (Italic Tag)**:   1. <i>: In nghiêng văn bản. 2. <em>: In nghiêng và nhấn mạnh văn bản.   **Thẻ gạch chân (Underline Tag)**: <u>: Gạch chân văn bản.  **Thẻ gạch ngang (Strikethrough Tag)**:   1. <s> hoặc <strike>: Gạch ngang văn bản. 2. <del>: Gạch ngang văn bản, thường dùng để chỉ văn bản bị xóa.   **Thẻ chỉ số trên (Superscript Tag)**: <sup>: Tạo chỉ số trên (ví dụ: a<sup>2</sup>).  **Thẻ chỉ số dưới (Subscript Tag)**: <sub>: Tạo chỉ số dưới (ví dụ: H<sub>2</sub>O).  **Thẻ đánh dấu (Mark Tag)**: <mark>: Làm nổi bật văn bản bằng cách tô nền.  **Thẻ nhỏ (Small Tag)**: <small>: Làm cho văn bản nhỏ hơn so với kích thước bình thường. |
| 22 | Thẻ div dùng để làm gì? <div> khác gì <span> | Thẻ <div> và thẻ <span> đều được sử dụng để nhóm các phần tử HTML lại với nhau, nhưng chúng có những điểm khác biệt quan trọng: Thẻ <div>  * **Định nghĩa**: Thẻ <div> (viết tắt của “division”) được sử dụng để tạo ra các khối (block) trong trang web. * **Công dụng**: Thường dùng để nhóm các phần tử HTML lại với nhau thành một khối lớn, giúp dễ dàng quản lý và định dạng bằng CSS. * **Đặc điểm**: Là phần tử khối (block-level element), nghĩa là nó chiếm toàn bộ chiều rộng của trang và bắt đầu trên một dòng mới.  Thẻ <span>  * **Định nghĩa**: Thẻ <span> được sử dụng để nhóm các phần tử HTML lại với nhau trong một dòng (inline). * **Công dụng**: Thường dùng để áp dụng các kiểu CSS hoặc JavaScript cho một phần nhỏ của văn bản hoặc nội dung trong một dòng. * **Đặc điểm**: Là phần tử nội tuyến (inline-level element), nghĩa là nó không bắt đầu trên một dòng mới và chỉ chiếm không gian cần thiết cho nội dung của nó.  Sự khác biệt chính  * **Khối vs. Nội tuyến**: <div> là phần tử khối, trong khi <span> là phần tử nội tuyến. * **Cách sử dụng**: <div> thường được dùng để tạo cấu trúc và bố cục trang web, còn <span> thường được dùng để định dạng và thao tác với các phần nhỏ của nội dung. |
| 23 | URL là gì? Phân biệt url tuyệt đối và url tương đối? | **URL** (Uniform Resource Locator) là địa chỉ web mà bạn sử dụng để truy cập vào các tài nguyên trên internet, chẳng hạn như trang web, hình ảnh, hoặc video. URL bao gồm các thành phần như giao thức (http, https), tên miền (domain), và đường dẫn (path) đến tài nguyên cụ thể.  **URL tuyệt đối** và **URL tương đối** là hai cách để định vị tài nguyên trên web:  **URL tuyệt đối**:   * Chứa toàn bộ đường dẫn từ giao thức đến tài nguyên. * Ví dụ: <https://www.example.com/images/photo.jpg> * URL tuyệt đối luôn chỉ đến một vị trí cụ thể trên internet, bất kể bạn đang ở đâu.   **URL tương đối**:   * Chỉ chứa đường dẫn từ vị trí hiện tại đến tài nguyên. * Ví dụ: images/photo.jpg * URL tương đối phụ thuộc vào vị trí hiện tại của tài liệu đang tham chiếu. Nếu di chuyển tài liệu này, URL tương đối có thể thay đổi theo |
| 24 | Siêu liên kết là gì? | **Siêu liên kết** (hay **hyperlink**) là một đoạn mã hoặc một phần tử trên trang web mà người dùng có thể nhấp vào để điều hướng đến một địa chỉ khác. Địa chỉ này có thể là một trang web khác, một phần khác của cùng trang, hoặc một tệp tin cụ thể |
| 25 | Muốn liên kết đến một phần trên cùng trang thì làm thế nào? | Để tạo liên kết đến một phần cụ thể trên cùng trang, bạn có thể sử dụng thuộc tính id trong HTML. Dưới đây là các bước cụ thể:   1. **Đánh dấu vị trí mục tiêu**:   Thêm thuộc tính id vào phần tử mà bạn muốn liên kết đến.  <h2 id="section1">Phần 1</h2>   1. **Tạo liên kết đến vị trí đó**:   Sử dụng thẻ <a> với thuộc tính href trỏ đến id của phần tử mục tiêu.  <a href="#section1">Chuyển đến Phần 1</a>  Khi người dùng nhấp vào liên kết, trình duyệt sẽ tự động cuộn đến phần tử có id tương ứng trên cùng trang |
| 26 | Muốn liên kết đến một phần trên trang khác thì làm thế nào? | Để tạo liên kết đến một phần cụ thể trên một trang khác, bạn cần sử dụng thuộc tính id để đánh dấu vị trí mục tiêu trên trang đích và sau đó tạo liên kết với đường dẫn đầy đủ đến trang đó kèm theo id của phần tử mục tiêu. Dưới đây là các bước cụ thể:   1. **Đánh dấu vị trí mục tiêu trên trang đích**:   Thêm thuộc tính id vào phần tử mà bạn muốn liên kết đến trên trang đích.  <!-- Trên trang đích (target-page.html) -->  <h2 id="section1">Phần 1</h2>   1. **Tạo liên kết trên trang hiện tại**:   Sử dụng thẻ <a> với thuộc tính href trỏ đến URL của trang đích và id của phần tử mục tiêu.  <!-- Trên trang hiện tại -->  <a href="/target-page.html#section1">Chuyển đến Phần 1 trên trang khác</a>  Khi người dùng nhấp vào liên kết này, trình duyệt sẽ mở trang đích và tự động cuộn đến phần tử có id tương ứng |
| 27 | Phân biệt internal link và external link? | **Internal link** và **external link** là hai loại liên kết quan trọng trong việc xây dựng và tối ưu hóa trang web:  **Internal link** (liên kết nội bộ):   * 1. **Định nghĩa**: Là các liên kết từ một trang này sang một trang khác trong cùng một website.   2. **Mục đích**: Giúp người dùng dễ dàng điều hướng và khám phá các nội dung liên quan khác trên cùng một trang web. Đồng thời, chúng giúp các công cụ tìm kiếm hiểu cấu trúc và thứ bậc của trang web.   3. **Ví dụ**: Trên một trang blog, liên kết từ bài viết này sang bài viết khác trên cùng blog.   **External link** (liên kết ngoài):   * 1. **Định nghĩa**: Là các liên kết từ một trang web này đến một trang web khác, không thuộc cùng một tên miền.   2. **Mục đích**: Cung cấp thêm thông tin hoặc tài nguyên từ các nguồn bên ngoài, giúp tăng độ tin cậy và uy tín của trang web. Các liên kết này cũng có thể giúp cải thiện thứ hạng SEO nếu được liên kết từ các trang web có uy tín.   **Ví dụ**: Một bài viết trên trang web của bạn liên kết đến một bài viết trên Wikipedia hoặc một trang web khác (Youtube, Facebook,...). |
| 28 | Có mấy loại danh sách? | Trong HTML, có 3 loại danh sách chính:  **Danh sách không sắp xếp (unordered list)**:   * 1. Sử dụng thẻ <ul>.   2. Các mục trong danh sách được đánh dấu bằng các ký hiệu như chấm tròn, hình vuông, hoặc hình tròn rỗng.   3. Ví dụ:   <ul>  <li>Mục 1</li>  <li>Mục 2</li>  <li>Mục 3</li>  </ul>  **Danh sách sắp xếp (ordered list)**:   * 1. Sử dụng thẻ <ol>.   2. Các mục trong danh sách được đánh dấu bằng số thứ tự, chữ cái, hoặc số La Mã.   3. Ví dụ:   <ol>  <li>Mục 1</li>  <li>Mục 2</li>  <li>Mục 3</li>  </ol>  **Danh sách mô tả (description list)**:   * 1. Sử dụng thẻ <dl> cho danh sách, <dt> cho thuật ngữ, và <dd> cho mô tả của thuật ngữ đó.   VD  <dl>          <dt>HTML</dt>          <dd>Ngôn ngữ đánh dấu siêu văn bản</dd>          <dt>CSS</dt>          <dd>Ngôn ngữ định kiểu cho trang web</dd>          <dd>Ngôn ngữ định kiểu cho trang web</dd>          <dd>Ngôn ngữ định kiểu cho trang web</dd>      </dl> |
| 29 | Muốn hiển thị danh sách có thứ tự bắt đầu C. D. E… thì làm thế nào? | Để hiển thị danh sách có thứ tự bắt đầu từ C, D, E,… trong HTML, có thể sử dụng thuộc tính start và type của thẻ <ol>. Dưới đây là ví dụ cụ thể:  <ol type="A" start="3">  <li>Mục 1</li>  <li>Mục 2</li>  <li>Mục 3</li>  </ol>  Trong ví dụ này:   * type="A": Đặt kiểu danh sách là chữ cái in hoa (A, B, C,…). * start="3": Bắt đầu danh sách từ chữ cái thứ ba trong bảng chữ cái, tức là C. |
| 30 | Form hay biểu mẫu dùng để làm gì? | **Form** hay **biểu mẫu** là công cụ dùng để nhập hoặc hiển thị thông tin một cách tiện lợi trong nhiều ứng dụng khác nhau. Biểu mẫu giúp người dùng thực hiện các thao tác như nhập liệu, lọc dữ liệu, và điều khiển thi hành chương trình.  Trong lập trình web, biểu mẫu thường được sử dụng để thu thập dữ liệu từ người dùng, chẳng hạn như đăng ký tài khoản, tìm kiếm thông tin, hoặc gửi các yêu cầu khác. Các biểu mẫu này thường bao gồm các phần tử như ô nhập liệu (input fields), nút bấm (buttons), và các tùy chọn (checkboxes, radio buttons). |
| 31 | Thuộc tính action của form dùng để làm gì? | Action có giá trị là URL quy định nơi dữ liệu được gửi đến |
| 32 | Phân biệt 2 method là get và post? | |  |  | | --- | --- | | GET | POST | | Dữ liệu được gửi đi sẽ hiển thị trên thanh địa chỉ | Dữ liệu không được gửi đi sẽ hiển thị trên thanh địa chỉ | | Không nên dùng để gửi các dữ liệu nhạy cảm như sđt, password | Dùng để gửi dữ liệu nhạy cảm, cần tính bảo mật. | | Gửi lượng dữ liệu nhỏ | Gửi lượng dữ liệu lớn | |
| 33 | Phân biệt các nút bấm có type là: button, submit, reset? | - button: dùng để thực hiện các hành động tùy chỉnh khi nhấn nút (không dùng để gửi dữ liệu khi nhấn)  - submit: gửi dữ liệu của form đến máy chủ thông qua URL được chỉ định trong action khi nhấn nút  - reset: đặt lại tất cả các giá trị trong form về trạng thái ban đầu (xóa toàn bộ dữ liệu trong form) |
| 34 | Muốn submit form đến trang CodeGym.vn thì làm thế nào? | <form action="CodeGym.vn" method="post">  <input type="text">  <button type="submit">Submit</button>  </form> |
| 35 | Nếu muốn nối 3 ô trên cùng 1 hàng thì làm thế nào? | <table>  <tr>  <td colspan="3">Nối 3 ô này lại</td>  </tr>  <tr>  <td>Ô 1</td>  <td>Ô 2</td>  <td>Ô 3</td>  </tr>  </table> |
| 36 | Nếu muốn nối 2 ô trên cùng 1 cột thì làm thế nào? | <table border="1">  <tr>  <td rowspan="2">Nối 2 ô này lại</td>  <td>Ô 1</td>  </tr>  <tr>  <td>Ô 2</td>  </tr>  </table> |
| 37 | Phân biệt thuộc tính cellpadding và thuộc tính cellspacing? | Giống: đều được dùng để điều chỉnh khoảng cách trong bảng (table)  - cellpadding: xác định khoảng cách giữa nội dung trong ô với đường viền.  - cellspacing: xác định khoảng cách giữa các ô  VD  <table border="1" cellpadding="10" cellspacing="10"> |
| 38 | Trình bày các cách nhúng javascript vào 1 tài liệu html? | 1. Nhúng trực tiếp (inline)   Vd: <button onclick="alert('Hello, world!')">Click me</button>   1. Nhúng nội bộ (internal)   Vd:  <body>  <button onclick="showMessage()">Click me</button>  <script>  function showMessage() {  alert('Hello, world!');  }  </script>  </body>   1. Nhúng bên ngoài (external)   Vd:  <body>  <button onclick="showMessage()">Click me</button>  <script src=”script.js”></script>  </body> |
| 39 | Phát biểu: “JavaScript là một ngôn ngữ script dựa trên đối tượng nhằm phát triển các ứng dụng Internet dựa trên client và server” là đúng hay sai? Giải thích? | Đúng  - Phía client: JavaScript chạy trên trình duyệt của người dùng, giúp tạo ra các trải nghiệm người dùng phong phú và tương tác.  - Phía server: Với sự ra đời của Node.js, JavaScript cũng có thể chạy trên máy chủ, cho phép xây dựng các ứng dụng server-side hiệu quả.  Server-side là phần xử lý diễn ra trên máy chủ trong quá trình xử lý yêu cầu từ phía người dùng (client). Khi một client (thường là trình duyệt hoặc ứng dụng) gửi yêu cầu đến máy chủ, máy chủ sẽ xử lý yêu cầu này, thường bao gồm việc:  - Truy xuất dữ liệu từ cơ sở dữ liệu.  - Xử lý logic nghiệp vụ.  - Xây dựng và trả về nội dung động cho client. |
| 40 | Phát biểu “JavaScript phân biệt chữ hoa chữ thường ” đúng hay sai? | Đúng |
| 41 | Trình bày về quy tắc CamelCase và camelCase? | - CamelCase: Cách đặt tên mà từ đầu tiên bắt đầu bằng chữ cái viết hoa, và các từ tiếp theo cũng viết hoa chữ cái đầu, thường dùng cho tên lớp hoặc hàm.  - camelCase (Lower Camel Case): Cách đặt tên mà từ đầu tiên bắt đầu bằng chữ cái thường, các từ tiếp theo viết hoa chữ cái đầu, thường dùng cho tên biến, thuộc tính, hoặc phương thức. |
| 42 | Biến là gì? Phân biệt biến cục bộ (local) và biến toàn cục (global)? | Biến là một tên được đặt để lưu trữ dữ liệu trong bộ nhớ, giúp lập trình viên dễ dàng truy cập, sử dụng và thay đổi dữ liệu trong chương trình.   |  |  | | --- | --- | | Biến cục bộ | Biến toàn cục | | Được khai báo bên trong một khối lệnh (hàm, phương thức, vòng lặp,...) và chỉ có tác dụng trong khối lệnh đó, chỉ tồn tại trong thời gian thực thi khối lệnh đó, tiêu tốn ít tài nguyên hơn do chỉ tồn tại trong phạm vi cụ thể | Được khai báo ngoài mọi hàm hoặc phương thức, có phạm vi toàn chương trình. Có thể được truy cập ở mọi nơi trong chương trình. Tồn tại trong suốt thời gian chạy chương trình. Tiêu tốn nhiều tài nguyên hơn vì tồn tại lâu hơn. | |
| 43 | Phân biệt: var, let, const? | |  |  |  | | --- | --- | --- | | var | let | const | | Có phạm vi trong hàm hoặc toàn cục, có thể tái khai báo và thay đổi giá trị. | Có phạm vi trong khối lệnh, không thể tái khai báo, có thể thay đổi giá trị. | Có phạm vi trong khối lệnh, không thể tái khai báo lẫn thay đổi giá trị. | |
| 44 | Phát biểu “JavaScript là ngôn ngữ định kiểu yếu hay động” nghĩa là gì? Trình bày? | Có nghĩa là JavaScript có cách quản lý kiểu dữ liệu linh hoạt, không chặt chẽ như một số ngôn ngữ khác (như Java, C++).   1. Định kiểu yếu (Weakly Typed):   - Trong JavaScript, một biến không cần khai báo kiểu dữ liệu rõ ràng khi tạo, và kiểu dữ liệu của biến có thể tự động chuyển đổi (type coercion) trong các phép toán hoặc ngữ cảnh khác.  - JavaScript cho phép thực hiện các phép toán giữa các giá trị khác kiểu mà không cần thông báo lỗi.   1. Định kiểu động (Dynamically Typed):   - Trong JavaScript, kiểu dữ liệu của một biến có thể thay đổi trong thời gian chạy mà không cần khai báo lại.  - Khi gán giá trị mới cho biến, kiểu của biến sẽ tự động cập nhật theo kiểu của giá trị đó. |
| 45 | Nêu tên các kiểu dữ liệu nguyên thủy trong JavaScript? | Number  BigInt  String  Boolean  Undefined  Null  Symbol |
| 46 | Trình bày về toán tử typeof? Khi khai báo “let x=10;” thì x có kiểu là gì? | Toán tử typeof được sử dụng để kiểm tra kiểu dữ liệu của một giá trị hoặc biến trong JavaScript.  “let x=10;” x là kiểu Number |
| 47 | Trình bày các cách đưa thông báo trong JavaScript? | alert(), confirm(), prompt(), console.log(), console.warn() |
| 48 | Trình bày về các loại toán tử trong JavaScript? | 1. Toán tử số học: +, -, \*, /, %, ++, --  2. Toán tử so sánh: ==, ===, !=, !==, >, >=, <, <=  3. Toán tử logic: &&, ||, !  4. Toán tử gán: =, +=, -=, \*=, /=, %=  5. Toán tử điều kiện: ? :  6. Toán tử loại: typeof, instanceof (trả ra true, false) |
| 49 | Phân biệt giữa 2 toán tử “==” và “===” | Toán tử == thực hiện so sánh giá trị, nhưng không so sánh kiểu dữ liệu.  Toán tử === so sánh cả giá trị và kiểu dữ liệu. |
| 50 | Phân biệt giữa 2 toán tử && và || trong JavaScript? | Toán tử && trả về true nếu tất cả các điều kiện đều đúng.  Toán tử || trả về true nếu ít nhất một trong các điều kiện là đúng. |
| 51 | Kết quả biểu thức +0===-0 là true hay false? | true |
| 52 | Trình bày về toán tử ++, -- và – trong JavaScript? | ++ tăng giá trị biến lên 1  -- giảm giá trị biến đi 1  - là toán tử trừ hoặc đảo dấu giá trị |
| 53 | Phân biệt ++a và a++? | ++a: Tăng giá trị của a trước, sau đó sử dụng giá trị đã tăng.  a++: Sử dụng giá trị hiện tại của a trước, sau đó mới tăng giá trị của a. |
| 54 | Câu lệnh điều kiện/rẽ nhánh là gì? | Là một cấu trúc trong lập trình dùng để kiểm tra các điều kiện và thực hiện các khối lệnh khác nhau dựa trên kết quả của điều kiện đó |
| 55 | Có những loại câu lệnh điều kiện nào? | If, if-else, if-else if-else, switch, toán tử ba ngôi |
| 56 | Trình bày về câu lệnh if? | Kiểm tra một điều kiện cụ thể. Nếu điều kiện đúng (true), khối lệnh bên trong sẽ được thực thi.  Vd:  if (điều\_kiện) {  // Khối lệnh thực hiện nếu điều kiện đúng  } |
| 57 | Trình bày về câu lệnh switch? | Dùng để kiểm tra giá trị của một biến với nhiều trường hợp cụ thể. Hiệu quả hơn if-else trong một số tình huống.  Vd:  switch (biến) {  case giá\_trị1:  // Khối lệnh nếu biến == giá\_trị1  break;  case giá\_trị2:  // Khối lệnh nếu biến == giá\_trị2  break;  default:  // Khối lệnh mặc định nếu không có trường hợp nào đúng  break;  } |
| 58 | Tại sao lại nói else là một tùy chọn (optional) của câu lệnh if? | Cấu trúc cơ bản của if không bắt buộc phải có else và else chỉ được sử dụng khi có nhu cầu xử lý điều kiện sai. |
| 59 | Khi nào nên sử dụng câu lệnh switch? | Câu lệnh switch được sử dụng khi cần kiểm tra một giá trị cụ thể của một biến và thực hiện các hành động khác nhau tùy thuộc vào giá trị đó. So với các câu lệnh if-else, switch thường gọn gàng và dễ đọc hơn trong một số trường hợp |
| 60 | Câu lệnh break dùng để làm gì? Nếu một case mà không có break thì chương trình chạy thế nào? | Câu lệnh break được sử dụng để:  - Kết thúc sớm vòng lặp hoặc cấu trúc điều kiện: Thoát khỏi vòng lặp hoặc khối switch ngay khi gặp break, bỏ qua phần còn lại của khối.  - Ngăn chặn việc thực hiện tiếp các trường hợp (case) khác trong switch: Trong cấu trúc switch, break giúp dừng kiểm tra ngay sau khi khối lệnh của một case được thực thi.  \* Khi một case trong switch không có câu lệnh break, chương trình sẽ tiếp tục thực thi các câu lệnh trong tất cả các case tiếp theo, bất kể điều kiện của chúng có khớp hay không. Hiện tượng này gọi là fall-through (chảy qua) |
| 61 | Mệnh đề default có bắt buộc trong câu lệnh switch? | Không bắt buộc |
| 62 | Trình bày về toán tử 3 ngôi? | Là một cú pháp ngắn gọn để viết câu lệnh điều kiện if-else. Thay vì sử dụng nhiều dòng để kiểm tra và gán giá trị, bạn có thể sử dụng toán tử này để viết mã ngắn gọn hơn. |
| 63 | Trình bày về sự kiện và trình quản lý sự kiện trong javascript? | - Sự kiện (event) trong JavaScript là những hành động hoặc tương tác xảy ra trên một trang web hoặc trình duyệt. Những sự kiện này có thể được kích hoạt bởi người dùng (như nhấp chuột, nhấn phím, di chuyển chuột) hoặc bởi trình duyệt (như tải trang, thay đổi kích thước cửa sổ).  - Trình quản lý sự kiện: addEventListener, removeEventListener |
| 64 | Sự kiện onchange phát sinh khi nào? | Sự kiện onchange xảy ra khi giá trị của một phần tử đầu vào (input) thay đổi và người dùng rời khỏi (focus-out) hoặc xác nhận giá trị. |
| 65 | Sự kiện onclick phát sinh khi nào? | Sự kiện onclick xảy ra khi người dùng nhấp chuột (click) vào một phần tử trên trang web. |
| 66 | Muốn hiển thị 1 hộp thoại xác nhận “Bạn chắc chắn xóa không?” với người dùng khi bấm vào nút delete thì làm thế nào? | <button id="deleteButton">Delete</button>  <script>  const deleteButton = document.getElementById('deleteButton');  deleteButton.addEventListener('click', function () {  // Hiển thị hộp thoại xác nhận  const userConfirmed = confirm('Bạn chắc chắn xóa không?');    if (userConfirmed) {  // Xử lý hành động xóa  alert('Xóa thành công!');  } else {  // Hủy bỏ hành động  alert('Hành động bị hủy.');  }  });  </script> |
| 67 | Phân biệt onmousevover và onmouseout, onmousemove? | - onmouseover: Sự kiện này sẽ xảy ra cả khi di chuyển chuột vào các phần tử.  - onmouseout: Xảy ra khi con trỏ chuột di chuyển ra khỏi vùng của một phần tử.  - onmousemove: Xảy ra khi con trỏ chuột di chuyển trong vùng của một phần tử. |
| 68 | Phân biệt onkeyup, onkeydown và onkeypress? | - onkeydown: Xảy ra khi một phím trên bàn phím được nhấn xuống.  - onkeyup: Xảy ra khi một phím trên bàn phím được thả ra.  - onkeypress (không còn khuyến nghị): Xảy ra khi một phím chữ cái trên bàn phím được nhấn xuống. |
| 69 | Phân biệt onfocus và onblur? | onfocus và onblur là hai sự kiện thường được sử dụng trong JavaScript để xử lý tương tác của người dùng với các phần tử nhập liệu (form elements).  - onfocus: Khi một phần tử nhận được tiêu điểm (focus), thường là khi người dùng nhấp chuột vào hoặc di chuyển tới phần tử bằng phím Tab. Thường sử dụng với các phần tử có thể nhận tiêu điểm, như: <input>, <textarea>, <button>, <select>. Có thể dùng để thay đổi giao diện phần tử khi nó được chọn, ví dụ: làm nổi bật trường nhập liệu.  - onblur: Khi một phần tử mất tiêu điểm (blur), thường là khi người dùng nhấn chuột ra ngoài hoặc di chuyển tiêu điểm đến phần tử khác. Thường dùng để kiểm tra và xử lý dữ liệu nhập sau khi người dùng rời khỏi phần tử. Hữu ích cho việc kiểm tra tính hợp lệ của dữ liệu nhập (validation).  Vd  <input type="text"      onfocus="this.style.backgroundColor='lightblue'"      onblur="this.style.backgroundColor='white'"      placeholder="Nhấn vào và nhập dữ liệu"> |
| 70 | Sự kiện onload phát sinh khi nào? | Sự kiện onload được kích hoạt khi một tài nguyên (resource) như trang web, hình ảnh, hoặc tài liệu được tải xong hoàn toàn. Nó đảm bảo rằng toàn bộ nội dung hoặc tài nguyên cần thiết đã sẵn sàng để sử dụng.  \* Đối với toàn bộ trang web:  - Khi tất cả nội dung của trang, bao gồm tài nguyên như hình ảnh, tập lệnh, CSS, iframe, và các tài nguyên con khác, đã được tải xong.  - Thường sử dụng trong thẻ <body> hoặc đối tượng window.  vd:  window.onload = function () {  console.log("Trang đã tải xong hoàn toàn!");  };  \* Đối với một tài nguyên cụ thể:  - Khi một tài nguyên cụ thể như hình ảnh, iframe, hoặc video được tải xong.  - Thường sử dụng cho các thẻ như <img>, <iframe>, hoặc trong JavaScript với đối tượng HTML tương ứng.  vd:  <img src="example.jpg" onload="console.log('Hình ảnh đã tải xong!')"> |
| 71 | Phân biệt innerHTML và innerText? | Sử dụng innerHTML khi cần thay đổi hoặc thao tác nội dung chứa các thẻ HTML.  Sử dụng innerText khi chỉ cần làm việc với văn bản thuần, không yêu cầu xử lý thẻ HTML. |
| 72 | Nếu muốn vô hiệu hóa (disabled) một button khi click vào 1 checkbox thì làm thế nào? | <body>      <input type="checkbox" id="myCheckbox">      <br><br>      <button id="myButton">Submit</button>      <script>          // Lấy tham chiếu đến checkbox và button          const checkbox = document.getElementById("myCheckbox");          const button = document.getElementById("myButton");          // Lắng nghe sự kiện thay đổi trạng thái của checkbox          checkbox.addEventListener("change", function () {              // Nếu checkbox được chọn, vô hiệu hóa button              if (checkbox.checked) {                  button.disabled = true;              } else {                  button.disabled = false;              }          });      </script>  </body> |
| 73 | Phân biệt thuộc tính readonly và disabled của html? | readonly (chỉ đọc) được sử dụng để làm cho nội dung của một phần tử không thể thay đổi, nhưng người dùng vẫn có thể tương tác với nó, ví dụ: sao chép văn bản. Phần tử được gửi qua form bình thường.  disabled (bị vô hiệu hóa) được sử dụng để làm cho phần tử không thể tương tác hoặc truy cập được. Phần tử bị bỏ qua khi gửi form |
| 74 | Vòng lặp là gì? | Vòng lặp là một cấu trúc cơ bản trong lập trình, được sử dụng để thực thi một đoạn mã nhiều lần cho đến khi một điều kiện nào đó được thỏa mãn. |
| 75 | Nêu các loại vòng lặp cơ bản trong JavaScript? | for  while  do-while |
| 76 | Phân biệt vòng lặp for,while,do…while? | For: biết trước số lần lặp  While: không biết trước số lần lăp  do-while: lặp ít nhất 1 lần |
| 77 | Khi nào nên sử dụng vòng lặp do…while? | Khi biết trước số lần lặp |
| 78 | Khi nào nên sử dụng vòng lặp for? | Khi không biết trước số lần lặp và cần kiểm tra điều kiện trước khi lặp. |
| 79 | Khi nào nên sử dụng vòng lặp while? | Khi cần thực hiện ít nhất một lần trước khi kiểm tra điều kiện. |
| 80 | Yêu cầu người dùng nhập đi nhập lại giá trị của một số cho đến khi số đó là số chẵn thì có nên sử dụng vòng lặp không? Nếu có, nên sử dụng loại vòng lặp nào? | Nên sử dụng vòng lặp để đảm bảo chương trình chạy liên tục cho đến khi đạt điều kiện là nhập vào số chẵn. Vòng lặp do-while là lựa chọn phù hợp |
| 81 | Yêu cầu người dùng tính tổng các số chẵn từ 10 đến 100 thì nên dùng vòng lặp loại nào? | Nên dùng vòng lặp for vì đã biết trước số lần lặp  let sum = 10 + 100;  for (let index = 12; index < 100; index += 2) {    sum += index;  } |
| 82 | Trình bày về vòng for thiếu? | vòng for thiếu xảy ra khi một hoặc nhiều thành phần trong cấu trúc trên được bỏ trống nhưng vòng lặp vẫn hợp lệ. JavaScript cho phép điều này miễn là cú pháp chung của for được duy trì (phải có dấu chấm phẩy ; giữa các phần) |
| 83 | Nếu muốn hiển thị các cặp số nguyên dương có tổng là 10 thì viết thế nào? | let target = 10; // Tổng cần tìm  for (let i = 1; i < target; i++) { // Duyệt từ 1 đến 9  let j = target - i; // Tìm số còn lại trong cặp  if (j >= i) { // Đảm bảo số còn lại lớn hơn i để không lặp cặp  console.log(`(${i}, ${j})`);  }  } |
| 84 | Nếu muốn hiển thị hình tam giác vuông \* thì nên viết thế nào? | let n = 5; // Số dòng của tam giác  for (let i = 1; i <= n; i++) { // Dòng 1 -> dòng n  let row = ""; // Biến lưu chuỗi ký tự của từng dòng  for (let j = 1; j <= i; j++) { // Số lượng \* trong mỗi dòng  row += "\*"; // Thêm \*  }  console.log(row); // In ra dòng hiện tại  } |
| 85 | Trình bày về cách chạy của vòng for lồng nhau? | Khi sử dụng các vòng lặp for lồng nhau, vòng lặp bên trong sẽ chạy hết tất cả các lần lặp cho mỗi lần lặp của vòng lặp bên ngoài. |
| 86 | Phân biệt giữa câu lệnh break và continue? | break: dùng để thoát khỏi vòng lặp  continue: bỏ qua lần lặp hiện tại và tiếp tục lần lặp tiếp theo |
| 87 | Hiển thị các số không chia hết cho 5 từ 1-100 có sử dụng continue thì làm thế nào? | for(let i = 1; i <= 100; i++) {  if(i % 5 == 0) continue;  console.log(i);  } |
| 88 | Mảng là gì? | Mảng (Array) trong JavaScript là một cấu trúc dữ liệu dùng để lưu trữ một danh sách các phần tử |
| 89 | Trình bày cú pháp khai báo mảng trong JavaScript? | let array = [];  let array = new Array();  let array = Array.of(5); // Mảng chứa một phần tử là số 5 |
| 90 | Chỉ số của mảng là gì? Chỉ số của mảng xuất phát từ 0 hay 1? | Chỉ số mảng là số thứ tự của các phần tử trong mảng và xuất phát từ 0 |
| 91 | Kích thước của mảng là gì? | Kích thước của mảng là số lượng phần tử mà mảng có thể chứa. |
| 92 | Phân biệt push() và unshift()? | push(): Thêm một hoặc nhiều phần tử vào cuối mảng.  unshift(): Thêm một hoặc nhiều phần tử vào đầu mảng. |
| 93 | Phân biệt push() và pop()? | push(): Thêm một hoặc nhiều phần tử vào cuối mảng.  pop(): Loại bỏ phần tử cuối cùng của mảng và trả về phần tử đó. |
| 94 | Phân biệt shift() và unshift()? | shift(): Loại bỏ phần tử đầu tiên của mảng và trả về phần tử đó.  unshift(): Thêm một hoặc nhiều phần tử vào đầu mảng. |
| 95 | Nếu muốn xóa phần tử cuối cùng của mảng dùng hàm nào? | Pop() |
| 96 | Nếu xóa phần tử đầu tiên của mảng dùng hàm nào? | Shift() |
| 97 | Muốn sắp xếp mảng thì dùng hàm nào? | Sort() |
| 98 | Có một mảng gồm tên của các bạn học viên. Nếu muốn sắp xếp tên này theo thứ tự alphabet làm thế nào? | Sort() |
| 99 | Thao tác duyệt mảng là gì? Triển khai thao tác bằng code? | Thao tác duyệt mảng là quá trình lặp qua từng phần tử của mảng để thực hiện một số hành động nhất định, chẳng hạn như in ra giá trị, tính toán, hoặc thay đổi giá trị của các phần tử. Trong JavaScript, có thể duyệt mảng bằng nhiều cách khác nhau, bao gồm sử dụng vòng lặp for, while, forEach, và các phương thức khác như map, filter.  let arr = [1, 2, 3, 4, 5];  for (let i = 0; i < arr.length; i++) {  console.log(arr[i]);  } |
| 100 | Thao tác duyệt mảng 2 chiều? | let matrix = [  [1, 2, 3],  [4, 5, 6],  [7, 8, 9]  ];  for (let i = 0; i < matrix.length; i++) {  for (let j = 0; j < matrix[i].length; j++) {  console.log(matrix[i][j]);  }  } |
| 101 | Mảng 2 chiều cần mấy chỉ số để xác định duy nhất 1 phần tử? | Cần 2 số |
| 102 | Viết code tính tổng các phần tử của ma trận số (ý tưởng)? | function sumMatrix(matrix) {  let total = 0;  for (let row of matrix) {  for (let element of row) {  total += element;  }  }  return total;  } |
| 103 | Viết code tính tổng các ô trên cùng hàng của ma trận số (ý tưởng)? | function sumRow(matrix, rowIndex) {  let total = 0;  for (let element of matrix[rowIndex]) {  total += element;  }  return total;  } |
| 104 | Viết code tính tổng các ô trên cùng cột của ma trận số (ý tưởng)? | function sumColumn(matrix, columnIndex) {  let total = 0;  for (let row of matrix) {  total += row[columnIndex];  }  return total;  } |
| 105 | Hàm (function) là gì? | Hàm (function) là một đoạn mã trong lập trình được thiết kế để thực hiện một nhiệm vụ cụ thể. |
| 106 | Phân biệt giữa tham số hình thức (formal arguments) và tham số thực (actual arguments)? | Tham số hình thức: Là các biến được định nghĩa trong phần khai báo của hàm. Chúng đóng vai trò như các chỗ trống để nhận giá trị khi hàm được gọi.  Tham số thực (đối số): Là các giá trị hoặc biến thực tế được truyền vào hàm khi hàm được gọi. Chúng thay thế các tham số hình thức trong quá trình thực thi hàm. |
| 107 | Gọi hàm (invoke) là gì? | là quá trình thực hiện một hàm đã được định nghĩa trong chương trình. |