**BỘ CÂU HỎI PHỎNG VẤN MODULE 1**

**BOOTCAMP PREPARATION 2.0**

|  |  |
| --- | --- |
| **STT** | **Câu hỏi** |
| 1 | **Hệ thống các kiến thức đã học ở Module1**   |  | | --- | | - Học các học & quản lý công việc bằng Kanban | | - Mô tả thuật toán bằng Pseudo code & Flowchart | | - Git & HTML | | - HTML form & table | | - Tổng quan Js (cấu trúc điều kiện, cấu trúc lặp, mảng, hàm, lập trình hướng đối tượng) | | - Thuật toán | |
| 2 | **Giải thuật/Thuật toán là gì?**  Là các chỉ thị để giải quyết vấn đề   * Một tập hợp các bước thực hiện được xác định rõ ràng cho việc giải quyết vấn đề gọi là giải thuật (Algorithm hay gọi vắn tắt là algo ). * Một giải thuật (còn gọi là thuật toán) có thể được định nghĩa như là một thủ tục, công thức hay cách giải quyết vấn đề. Nó gồm một tập hợp các bước giúp đạt được lời giải. * Giải thuật chúng ta có là một tập hợp các bước được liệt kê dưới dạng ngôn ngữ đơn giản. |
| 3 | **So sánh việc trình bày giải thuật bằng pseudo-code và flowchart?**   |  |  |  | | --- | --- | --- | |  | **Pseudo Code** | **Flow chart** | |  | Không cần kiến thức về cách biểu diễn. Không có cấu trúc, dài dòng, khó hiểu | Quy định chặt chẽ từng kí hiệu nên thống nhất về cách dùng và dễ hiểu hơn | | **Bắt đầu / kết thúc** | Begin / End | Start / Stop | | **Cấu trúc** | IF … Then  ELSE … Then |  | |
| 4 | **Tại sao lại cần trình bày giải thuật trước khi bắt tay vào code?**   * Để nắm bắt vấn đề rõ hơn, có cái nhìn tổng quan hơn về dòng logic trong chương trình của mình. |
| 5 | **Tại sao lại nên sử dụng flowchart hơn pseudo-code khi thể hiện giải thuật của bài toán phức tạp?**   * Bởi vì các thông tin được trình bày dưới dạng flowchart vô cùng ngắn gọn, hiệu quả. * Diễn giải các công việc tưởng chừng như phức tạp thành từng quy trình nghiên cứu, hoạt động làm việc theo trình tự rõ ràng. * Từ đó dễ dàng đánh giá, phân tích và chọn ra quy trình tối ưu nhất. * Một lưu đồ là một hình ảnh minh hoạ cho giải thuật. * Nó vẽ ra biểu đồ của luồng chỉ thị hay những hoạt động trong một tiến trình. * Mỗi hoạt động như vậy được biểu diễn qua những ký hiệu. |
| **6** | **Tại sao giải thuật lại cần hữu hạn các bước thực thi?**  **IPO: Input – Process – Output**   * Input: Đầu vào * Process: Các bước xử lý * Output: Đầu ra – kết quả trả về |
| 7 | **Trình bày ý nghĩa các ký hiệu sử dụng khi vẽ flowchart?**  C:\Users\nguye\OneDrive\Máy tính\laptrinh02.png |
| 8 | **Vẽ giải thuật giải phương trình bậc nhất?**  D:\Downloads\img_4720.jpg |
| 9 | **Bạn biết gì về VCS – Version Control System (hệ thống quản lý phiên bản)?**  Là hệ thống quản lí phiên bản mã nguồn, hỗ trợ quản lí toàn bộ quá trình của dự án   * Khôi phục lại phiên bản cũ của các file * Khôi phục lại phiên bản cũ của toàn bộ dự án * Xem lại các thay đổi đã được thực hiện theo thời gian * Xem ai là người thực hiện thay đổi cuối cùng có thể gây ra sự cố * Khôi phục lại các file vô tình xoá mất |
| 10 | **Phân biệt hệ thống quản lý mã nguồn tập trung và hệ thống quản lý mã nguồn phân tán?**   |  |  | | --- | --- | | Quản lí mã nguồn tập trung | Quản lí mã nguồn phân tán | | Dữ liệu từ các Local repo được đẩy lên lưu trữ ở hệ thống quản lý chung, gọi là Remote repo. Nếu hệ thống ở Remote repo bị sập thì sẽ không có khu vực khác để lưu trữ dữ liệu | Lưu trữ dữ liệu không chỉ ở Remote repo mà còn ở các Local repo khác. Do đó, khi Remote repo bị sập thì dữ liệu vẫn có thể được phục hồi từ các Local repo | |
| 11 | **Git là gì? Tại sao cần sử dụng git?**   * Git là một cách thức (mô hình, giao thức, công cụ, qui tắc) để lưu trữ mã nguồn và hỗ trợ cộng tác trong quá trình phát triển ứng dụng. * Git là hệ thống điều khiển phiên bản (VCS) theo hình thức phân tán |
| 12 | Git thuộc hệ thống quản lý mã nguồn tập trung hay **phân tán**?   * Git là hệ thống điều khiển phiên bản (VCS) theo hình thức phân tán |
| 13 | **Repository là gì? Phân biệt LocalRepository và RemoteRepository?**   * **Repository**: Là nơi chứa mã nguồn (repository có nghĩa là "kho chứa"), tức là nơi mà chúng ta lưu trữ các file, các đoạn mã được viết ra trong suốt quá trình phát triển dự án. Có 2 loại Repository là Local Repository và Remote Repository. * **Local Repository**: Là nơi lưu trữ mã nguồn trên máy tính của lập trình viên. * **Remote Repository**: Là nơi lưu trữ mã nguồn ở trên mạng (chẳng hạn là mạng Internet hay là mạng nội bộ của một công ty). Để có Remote Repository thì chúng ta cần có một máy chủ và cài Git Server lên đó. Remote Repository (có nghĩa là "kho chứa từ xa") là nơi mà cả Bob và James đều có thể truy cập được để tải mã nguồn về hoặc đưa mã nguồn lên đó. * Thường được gọi ngắn gọn là repo * Repository là nơi chứa toàn bộ mã nguồn * Repository bao gồm toàn bộ các file và lịch sử của các file đó * Repository chứa tất cả các commit * **Có 2 loại** repository: * Local Repository: Ở trên máy của lập trình viên * Remote Repository: Ở trên một máy chủ chia sẻ (chẳng hạn như GitHub) |
| 14 | **Đồng bộ giữa các repository (local và remote) để làm gì?**   * Ngoài Repository chung (chính là Remote Repository) thì Bob và James đều có các Repository riêng của mình (chính là Local Repository), do đó phải có cơ chế để đồng bộ giữa các Repository này với nhau, tức là giữ cho mã nguồn ở các Repository này giống hệt nhau. Hay nói cách khác, tất cả các Repository này là những bản sao của nhau. * Giúp cho các repo luôn luôn có bản sao mới nhất |
| 15 | **Thao tác push của git để làm gì? Cú pháp câu lệnh?**   * Đẩy mã nguồn từ Local Repository lên Remote Repository. * Cú pháp: git.push -u origin main (với lần đầu push source lên github) * Cú pháp: git push origin main (với các lần cập nhật sau đó) |
| 16 | **Thao tác pull của git để làm gì?**   * Cập nhật mã nguồn từ một Remote Repository về Local Repository |
| 17 | **Thao tác clone của git để làm gì? Cú pháp câu lệnh?**   * Sao chép một Remote Repository về máy của lập trình viên |
| 18 | **Nêu tên và ý nghĩa các câu lệnh cơ bản đã được học để làm việc với git?**   * **Git init: 🡺** khởi tạo một repo * **Git status 🡺** xem trạng thái local repo * **Git add . 🡺** Thêm tất cả thay đổi vào staging area * **Git commit –m “tên commit” 🡺** Lưu tất cả thay đổi từ Staging area vào Local repo * **Git remote add orgin [remote orgin url]** 🡺 Thêm kết nối từ local repo tới remote repo * **Git branch –M main 🡺** tạo nhánh main và setup nhánh chính là nhánh main * **Git push –u origin main 🡺** đẩy code từ local repo lên remote repo (-u lần đầu khi mới tạo local repo, lần 2 trả đi không cần –u khi push code) |
| 19 | **www là gì?** World Wide Web. Là mạng lưới toàn cầu mà người dùng có thể truy cập thông tin qua các thiết bị có kết nối mạng. |
| 20 | **HTML là gì?**   * HTML là ngôn ngữ được sử dụng để tạo ra các Webpage * HTML là viết tắt của chữ **Hypertext Markup Language** (Ngôn ngữ Đánh dấu Siêu Văn bản) * Đánh dấu: sử dụng các thẻ * Siêu văn bản: có thể đi đến văn bản khác thông qua các liên kết (link) |
| 21 | **Nêu tên các thẻ làm việc với văn bản trong tài liệu html?**   * Khai báo **<!DOCTYPE html>** quy định rằng đây là một tài liệu HTML5. * Phần tử **<html>** là phần tử gốc của trang HTML. * Phần tử **<head>** chứa thông tin mô tả về tài liệu hiện tại. * Phần tử **<title>** chứa tiêu đề của tài liệu. * Phần tử **<body>** chứa toàn bộ nội dung của trang web (những gì mà chúng ta nhìn thấy được trên trình duyệt). * Phần tử **<h1>** định nghĩa một tiêu đề lớn. * Phần tử **<p>** định nghĩa một đoạn văn. * Phần tử **<hr>** định nghĩa đường kẻ ngang. |
| 22 | **Thẻ div dùng để làm gì? <div> khác gì <span>**   * Thẻ **<div>** dùng để đánh dấu một khối (block) gồm nhiều thẻ khác trong khối, giúp chia tài liệu HTML thành nhiều phần riêng biệt * Thẻ **<span>** được sử dụng để chứa một đoạn văn bản ngắn mà lập trình viên muốn tạo kiểu riêng biệt. Thẻ div lại được sử dụng để chứa các kiểu dữ liệu khác nhau, không chỉ có văn bản chữ mà còn có hình ảnh, v.v... |
| 23 | **URL là gì? Phân biệt url tuyệt đối và url tương đối?**  Một đường dẫn liên kết đến website  **URL tuyệt đối**: một địa chỉ cụ thể trên trang web  **URL tương đối:** thay đổi tùy theo trang liên kết đang bật |
| 24 | **Siêu liên kết là gì?**   * Liên kết trong mã HTML còn được gọi là siêu liên kết (hyperlink). * Một siêu kiên kết là một văn bản hoặc một hình ảnh mà bạn có thể nhấp chuột vào và chuyển đến một tài liệu khác. * **Chú ý**: Ta thường dùng chữ để hiển thị liên kết. Nhưng ta cũng có thể dùng ảnh hoặc các thành phần HTML khác để hiển thị liên kết. |
| 25 | **Muốn liên kết đến một phần trên cùng trang thì làm thế nào?**  Tạo các bookmart(id)  Tạo liên kết trong (href)   * Thuộc tính **target** có: **\_self** - Mở liên kết trong tab/cửa sổ hiện tại của trình duyệt (Không cần khai báo cũng được vì đây là mặc định của thẻ <a>) |
| 26 | **Muốn liên kết đến một phần trên trang khác thì làm thế nào?** |
|  | * Thuộc tính **target** có: **\_blank** - Mở liên kết trong một cửa sổ hoặc một tab mới của trình duyệt |
| 27 | **Phân biệt internal link và external link?**   * **Internal link**: * Liên kết nội bộ trong website. * Một liên kết cục bộ khi trỏ đến địa chỉ trên trang web của chính mình. * Liên kết cục bộ có thuộc tính ***href*** là một địa chỉ URL tương đối (không có <http://www>...). * **External link:** Liên kết từ bên ngoài, có thể liên kết đến các trang bên ngoài bằng một địa chỉ URL đầy đủ hoặc đường dẫn tương đối so với trang web hiện tại. |
| 28 | **Có mấy loại danh sách?** **3 loại**  **ol (ordered list)**: Danh sách có thứ tự 1, 2, 3 hay a, b, c,…  **ul (unordered list)**: Danh sách không có thứ tự 1, 2, 3 hay a, b,  **dl:** tạo danh sách |
| 29 | **Muốn hiển thị danh sách có thứ tự bắt đầu C. D. E… thì làm thế nào?**  <ol ***type="A"*** **start=”3”**>  <li>HTML</li>  <li>Java</li>  <li>JavaScript</li>  <li>SQL</li>  </ol> |
| 30 | **Form hay biểu mẫu dùng để làm gì?**   * Biểu mẫu (form) là cơ chế cho phép người dùng nhập dữ liệu và gửi về server. Biểu mẫu thu thập thông tin người dùng. * Một form có thể bao gồm nhiều trường nhập dữ liệu khác nhau. |
| 31 | **Thuộc tính action của form dùng để làm gì?**   * Thẻ **<form>** được sử dụng để tạo một form * Thuộc tính ***action*** có giá trị là một URL, quy định nơi dữ liệu được gửi đến * Thuộc tính ***name*** quy định tên của form, sau đó có thể được dùng để truy xuất đến các giá trị trong form. |
| 32 | **Phân biệt 2 method là get và post?**   * **GET:**   + Gửi dữ liệu nhỏ.   + Không bảo mật thông tin. * **POST:**   + Gửi dữ liệu lớn.   + Bảo mật thông tin. |
| 33 | **Phân biệt các nút bấm có type là: button, submit, reset?**   * **Button**: Tạo một nút có thể gắn với các sự kiện của JS để xử lý thao tác của người dùng * **Submit**: Tạo một nút nhấn, khi người dùng nhấn vào nút submit thì dữ liệu sẽ được gửi lên server * **Reset**: Xóa các dữ liệu mà người dùng vừa mới nhập trong form, tất cả trường dữ liệu sẽ quay trở về dữ liệu mặc định trước đó. |
| 34 | **Muốn submit form đến trang CodeGym.vn thì làm thế nào?**   * Đưa đường dẫn vào thuộc tính **action**   <form action=”action\_page.php” method=”post”>  …..  </form> |
| 35 | **Nếu muốn nối 3 ô trên cùng 1 hàng thì làm thế nào?**  <tr>  <td ***colspan="3"***>  Hàng 1 cột 1 và Hàng 1 cột 2 và Hàng 1 cột 3  </td>  <td>Hàng 1 cột 4</td>  </tr> |
| 36 | **Nếu muốn nối 2 ô trên cùng 1 cột thì làm thế nào?**  <tr>  <td ***rowspan="2"***>Hàng 1 cột 1</td>  <td>Hàng 1 cột 3</td>  <td>Hàng 1 cột 3</td>  </tr> |
| 37 | **Phân biệt thuộc tính cellpadding và thuộc tính cellspacing?**   * **Cellspacing**: Tạo khoảng cách giữa viền trên và viền dưới của đường kẻ trong bảng * **Cellpadding**: Tạo khoảng cách giữa nội dung trong ô so với đường viền |
| 38 | **Trình bày các cách nhúng javascript vào 1 tài liệu html?**   * Viết mã Javascript bên trong thẻ <script></script> * Sử dụng file .js và nhúng vào trang web      * Sử dụng file .js từ bên ngoài (thư viện có sẵn).      * Viết trực tiếp mã JavaScript trong thẻ html |
| 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.**   * Nó là một trong những loại ngôn ngữ full-stack duy nhất được tính đến thời điểm hiện tại với nhiều tính năng nổi bật từ web client đến server, từ desktop đến mobile. * Dựa trên ý tưởng tạo ra các loại ngôn ngữ thông dịch tích hợp phù hợp với mọi trình duyệt để có thể nâng cao được các tính năng tương tác của người dùng đối với trang web. * Các script ở máy client có thể thực thi được trong trình duyệt trong khi các script ở máy server lại thực hiện trên server. |
| 40 | **Phát biểu “JavaScript phân biệt chữ hoa chữ thường ” đúng hay sai? Đúng** |
| 41 | **Trình bày về quy tắc CamelCase và camelCase?**   * Ký tự đầu tiên của từ đầu tiên viết thường những ký tự đầu tiên của những từ tiếp theo được viết hoa. |
| 42 | **Biến là gì? Phân biệt biến cục bộ (local) và biến toàn cục (global)?**   * **Biến** là một tên gọi được gắn cho một vùng nhớ chứa dữ liệu. * Dữ liệu được lưu trữ trong vùng nhớ của biến được gọi là **giá trị** * **Biến cục bộ** được khai báo bên trong một hàm. Biến cục bộ được tạo khi hàm bắt đầu thực thi và bị mất khi hàm kết thúc. * **Biến Toàn cục** được khai báo bên ngoài hàm. Biến toàn cục được tạo khi bắt đầu thực thi và bị mất khi chương trình kết thúc. |
| 43 | **Phân biệt: var, let, const?**   |  |  |  | | --- | --- | --- | | Var: toàn cục không nghiêm ngặt,dễ lỗi | Let: cục bộ nghiêm ngặt, biết được phạm vi sử dụng biến | Const: khai báo hằng giá trị không thay đổi | |
| 44 | **Phát biểu “JavaScript là ngôn ngữ định kiểu yếu hay động” nghĩa là gì? Trình bày?**   * JavaScript là một ngôn ngữ định **kiểu động**. * Điều đó nghĩa là không cần phải khai báo kiểu của các biến trước khi dùng. Kiểu sẽ được xác định tự động trong khi chương trình được thực thi. Điều đó cũng có nghĩa là một biến có thể chứa giá trị của các kiểu dữ liệu khác nhau. |
| 45 | **Nêu tên các kiểu dữ liệu nguyên thủy trong JavaScript?**  - **Sáu** kiểu nguyên thủy:   * Kiểu boolean * Kiểu của null * Kiểu của undefined * Kiểu số * Kiểu chuỗi * Kiểu Symbol (mới trong ECMAScript 6) |
| 46 | **Trình bày về toán tử typeof? Khi khai báo “let x=10;” thì x có kiểu là gì?**  Dùng để lấy về kiểu dữ liệu của một biến hoặc giá trị  Number |
| 47 | **Trình bày các cách đưa thông báo trong JavaScript?**   * Alert(); |
| 48 | **Trình bày về các loại toán tử trong JavaScript?** |
| 49 | **Phân biệt giữa 2 toán tử == và ===?**   |  |  | | --- | --- | | Dùng để kiểm tra 2 phần tử có bằng nhau không | | | ==: chỉ xét về kiểu giá trị | ===: nghiêm ngặt hơn xét về kiểu giá trị lẫn kiểu dữ liệu | |
| 50 | **Phân biệt giữa 2 toán tử && và || trong JavaScript?**   |  |  | | --- | --- | | **&&** | **||** | | sẽ trả về **TRUE** của nếu **tất cả** n điều kiện đều là true, và sẽ trả về **FALSE** nếu có tồn tại một toán hạng false | sẽ trả về **TRUE** nếu chỉ cần 1 trong các điều kiện là **TRUE**, bất kể các điều kiện còn lại trả về giá trị false | |
| 51 | **Kết quả biểu thức +0===-0 là true hay false?**  **True,** cùng là **Number** |
| 52 | **Trình bày về toán tử ++,-- và – trong JavaScript?**   * Chẳng hạn, ta có: i++ * Điều này tương tự với giá trị biểu thức i = i + 1, nghĩa là với một giá trị i ban đầu thì nó sẽ liên tục tạo ra giá trị i mới bằng cách cộng giá trị i trước đó với 1 * - :là phép trừ |
| 53 | **Phân biệt ++a và a++?**   * ++i tính luôn trên giá trị ban đầu. * i++ tạo ra bản copy của giá trị ban đầu. |
| 54 | **Câu lệnh điều kiện/rẽ nhánh là gì?**  **Rẽ nhánh luồng thực thi chương trình.** |
| 55 | **Có những loại câu lệnh điều kiện nào?**   * If dạng thiểu (if). * If đủ (if/else) * If lồng (nested-if) * If bậc thang (ladder-if): If – else if – else * Switch case  |  |  | | --- | --- | | **If - else** | **Switch – case** | | Điều kiện luôn trả về **true / false** | Biểu thức có thể trả về giá trị **chuỗi, số** | | Có thể so sánh >, >=, <, =<, ==, != | Chỉ so sánh == | | if trường hợp, logic phúc tạp | Số lần trường hợp **xác định (nhiều hơn 2 case)** | |
|  |  |
| 56 | **Trình bày về câu lệnh if?**   * **If dạng thiếu (if)** * Chỉ thực thi khi code **đúng**   If (đk) {  // code khi điều kiện đúng  }   * **If đủ (If – else)** * Thực hiện khi code trả về **đúng / sai**   If (đk) {  // code khi điều kiện đúng  } else {  // code khi điều kiện sai  }   * **If lồng (nested – if)**   If (diều kiện 1) {  If (điều kiện 2) {  // code khi đúng  }  else {  }  } else {  // code chạy khi sai  }   * **if bậc thang (ladder – if)**   If (điều kiện 1) {  } else if (điều kiện 2) {  } else if (điều kiện 3) {  } else {  } |
| 57 | **Trình bày về câu lệnh switch?**  Switch (biểu thức) {  Case 1:  //code  break;  Case 2:  //code  break;  ….  Default:  //code  break;  } |
| 58 | **Tại sao lại nói else là một tùy chọn (optional) của câu lệnh if?**  Vì nếu chúng ta không dùng else thì chúng ta vẫn có thể dùng if cho câu lệnh. |
| 59 | **Khi nào nên sử dụng câu lệnh switch?**   * Biểu thức có thể trả về giá trị chuỗi, số * Chỉ so sánh == * Số lần trường hợp xác định (nhiều hơn 2 case) |
| 60 | **Câu lệnh break dùng để làm gì? Nếu một case mà không có break thì chương trình chạy thế nào?**   * Từ khóa **break**, nó thoát ra khỏi khối switch. * Khi tìm thấy một case (trường hợp) phù hợp, và công việc được thực hiện, đó là lúc cần dừng lại. Sẽ không cần kiểm tra thêm các case khác nữa. * Nó sẽ thực thi cho đến khi gặp case có **break.** |
| 61 | **Mệnh đề default có bắt buộc trong câu lệnh switch?**  **Không** |
| 62 | **Trình bày về toán tử 3 ngôi?** |
| 63 | **Trình bày về sự kiện và trình quản lý sự kiện trong javascript?**  Sự kiện onchange, onmouseover và onmouseout, sự kiện onclick |
| 64 | **Sự kiện onchange phát sinh khi nào? (Câu 67)** |
| 65 | **Sự kiện onclick phát sinh khi nào? (Câu 67)** |
| 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?**   * confirm("Bạn chắc chắn xóa không?"); |
| 67 | **Phân biệt onmousevover và onmouseout, onmousemove?** |
| 68 | **Phân biệt onkeyup, onkeydown và onkeypress?**   * keydown: được gọi khi bạn nhấn xuống một key * keyup: được gọi khi bạn nhả key đó ra * keypress: được gọi khi bạn nhấn và giữ key |
| 69 | **Phân biệt onfocus và onblur?**   |  |  | | --- | --- | | onfocus | onblur | | Khởi tạo một hàm dạng cửa sổ hay frame khi di chuyển chuột vào cửa sổ | Sẽ được kích hoạt khi bạn rời nó | |
| 70 | **Sự kiện onload phát sinh khi nào?**   * Sự kiện onload có ý nghĩa rằng khi trình duyệt đã load xong mọi thứ (image, js, css) thì những đoạn code nằm bên trong đó mới được chạy. |
| 71 | **Phân biệt innerHTML và innerText?**   * **innerHTML:** Thêm ***nội dung*** vào bên trong Element. * **innerText:** * Sử dụng getter để lấy ra giá trị, thì nội dung lấy được sẽ giống như nội dung nhìn thấy trên trình duyệt. * Lấy luôn cả element con và bỏ qua thẻ HTML của element con, không có khoảng cách trước và sau. * Ảnh hưởng CSS trong thuộc tính. * Không trả về code của CSS và Js. |
| 72 | **Nếu muốn vô hiệu hóa (disabled) một button khi click vào 1 checkbox thì làm thế nào?**   * const button = document.querySelector("button"); * button.**setAttribute**("disabled", ""); |
| 73 | **Phân biệt thuộc tính readonly và disabled của html?**   |  |  | | --- | --- | | Readonly | Disabled | | Dùng để hiển thị không thể thay đổi | Vô hiệu hóa thẻ input, không thể nhấn hay sử dụng | |
| 74 | **Vòng lặp là gì?**   * Khối lệnh lặp lại nhiều lần. |
| 75 | **Nêu các loại vòng lặp cơ bản trong JavaScript?**   * For * While * do-while * forEach * for-in 🡺 key * for-of 🡺 value |
| 76 | **Phân biệt vòng lặp for,while,do…while?**   |  |  |  | | --- | --- | --- | | **for** | **while** | **Do - while** | | - Biết trước số lần lặp | - Không biết trước số lần lặp | - Thực thi phần thân ít nhất một lần | |  | | | |  | - Luôn kiểm tra điều kiện lặp trước khi thực thi | - Lần đầu không kiểm tra điều kiện lặp.  - Các bước lặp tiếp theo có kiểm tra điều kiện lặp | |
| 77 | **Khi nào nên sử dụng vòng lặp do…while?**  Thực thi phần thân ít nhất là một lần |
| 78 | **Khi nào nên sử dụng vòng lặp for?**  Khi đã biết trước số lần lặp |
| 79 | **Khi nào nên sử dụng vòng lặp while?**  Khi không biết trước số lần lặp |
| 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?**  Có nên sử dụng vòng lặp While |
| 81 | **Yêu cầu người dùng tính tổng các số chẵn từ 10 đến 100 thì nên dùng vòng lặp loại nào?**  Vòng lặp for vì biết trước số lần lặp |
| 82 | **Trình bày về vòng for thiếu?**  For(kiểm tra; điều kiện lặp; hoạt động sau những bước lặp){  //khối lệnh lặp  } |
| 83 | **Nếu muốn hiển thị các cặp số nguyên dương có tổng là 10 thì viết thế nào?**  for(let i = 1 ;i < 10; i++) {  for (let j = 1; j < 10; j++) {  If ( i + j ==10) {  console.log( i + “+” + j +”<br/>”);  }  }  } |
| 84 | **Nếu muốn hiển thị hình tam giác vuông \* thì nên viết thế nào?**  let hight = +prompt(“Nhập vào chiều cao của tam giác”)      hight = sc.nextInt();      for (i = 1; i <= hight; ++i) {        for (j = 1; j <= i; ++j) {          System.out.print("\* ");        }        System.out.println();      } |
| 85 | **Trình bày về cách chạy của vòng for lồng nhau?**  For lồng nhau thì đầu tiên chương trình sẽ check vòng for bên ngoài nếu điều kiện đúng thì sẽ nhảy vào trong thực hiện câu lệnh bên trong và cả vòng for bên trong. Chương trình tiếp tục check điều kiện vòng for thứ 2, nếu thỏa mãn sẽ thực thi câu lệnh bên trong vòng fr thứ 2, nếu không thỏa mãn thì không thực hiện câu lẹnh trong vòng for thứ 2 tiếp tục thực hiện các câu lệnh khác trong vòng for thứ nhất. Khi thực thi xong thực hiện hành động sau mỗi lần lặp rồi tiếp tục check điều kiện của vòng for thứ nhất. Lặp lại đến khi điều kiện của vòng for thứ nhất không thỏa mãn. |
| 86 | **Phân biệt giữa câu lệnh break và continue?**   * **Break:**    + Thoát ra khỏi vòng lặp hiện tại chứa nó   + Nhiều vòng lặp, break sẽ thoát ra khỏi vòng lặp gần nhất. * **Continue**: Bỏ qua lần lặp hiện tại, tiến tới bước 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?**   * Biến đặc biệt chứa nhiều giá trị. * Trong một mảng có thể nhiều kiểu dữ liệu (String, int, Boolean,…). Nhưng nên cùng kiểu. |
| 89 | **Trình bày cú pháp khai báo mảng trong JavaScript?**   * let arr = [2, 3, 4]; * let arr = new Array(2, 3, 4); |
| 90 | **Chỉ số của mảng là gì? Chỉ số của mảng xuất phát từ 0 hay 1?**   * Vị trí của các phần tử trong mảng. Chỉ số bắt đầu từ 0. |
| 91 | **Kích thước của mảng là gì?**  Số lượng các phần tử của mảng (length)  Array.length |
| 92 | **Phân biệt push() và unshift()?**   * Hàm **push()** sẽ thêm một phần tử ***mới vào phần cuối*** của mảng. * Hàm **unshift()** sẽ thêm một ***phần tử mới và phần đầu*** của mảng |
| 93 | **Phân biệt push() và pop()?**   * Hàm **pop()** sẽ ***lấy đi phần tử cuối cùng*** của mảng. |
| 94 | * **Phân biệt shift() và unshift()?** * Hàm **unshift()** sẽ thêm một ***phần tử mới và phần đầu*** của mảng * Hàm **shift()** sẽ **lấy đi phần tử đầu tiên** của 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?**   * Sử dụng vòng lặp For, chạy từ phần tử đầu tiên đến biến đếm (n-1). * Sử dụng phương thức ForEach, nhận tham số là 1 hàm trong đó có các phần tử của mảng. |
| 100 | **Thao tác duyệt mảng 2 chiều?**   * Dùng vòng lặp lồng nhau |
| 101 | **Mảng 2 chiều cần mấy chỉ số để xác định duy nhất 1 phần tử?**   * Cần có 2 chỉ số |
| 102 | **Viết code tính tổng các phần tử của ma trận số (ý tưởng)?**  for (let i =0; i< max.length; i++){  for(let y = 0;y < max[i].length; y++){  total += max[i][y];  }  } |
| 103 | **Viết code tính tổng các ô trên cùng hàng của ma trận số (ý tưởng)?**  for(let i = 0; i< max[0].length; i++ ){  let total = 0;  for (let y= 0; y < max.length; y++){  total += max[i][y];  console.log(“Sum for column” + y + “is” + total”);  } |
| 104 | **Viết code tính tổng các ô trên cùng cột của ma trận số (ý tưởng)?**  for(let i = 0; i< max[0].length; i++ ){  let total = 0;  for (let y = 0; y < max.length; y++){  total += max[i][y];  console.log(“Sum for row” + i+ “is” + total”);  }  } |
| 105 | **Hàm (function) là gì?**   * Hàm (function) là một nhóm các câu lệnh thực hiện một nhiệm vụ nhất định * Hàm là thuật ngữ được sử dụng phổ biến trong Lập trình hướng đối tượng. * Trong nhiều trường hợp khác, các tên gọi được sử dụng là phương thức (method) và thủ tục (procedure) |
| 106 | **Phân biệt giữa tham số hình thức (formal arguments) và tham số thực (actual arguments)?**   * Tham số (còn được gọi đầy đủ là tham số hình thức – formal parameter) là các biến được khai báo trong phần header * Khi gọi hàm thì giá trị của các biến này sẽ được truyền vào. Các giá trị này được gọi là tham số thực (actual parameter) hoặc đối số (argument) |
| 107 | **Gọi hàm (invoke) là gì?**   * Gọi (call hoặc invoke) phương thức là cách để thực thi một hàm đã được định nghĩa trước đó * Khi gọi hàm thì cần truyền đối số vào |
| 108 | **Phân biệt hàm gọi (calling function) và hàm được goi (called function)?**   |  |  | | --- | --- | | **calling function** | **called function** | | Hàm đang được sử dụng | Hàm đã sử dụng được rồi | |
| 109 | **Phân biệt truyền tham trị và truyền tham chiếu?**   |  |  | | --- | --- | | **Truyền tham trị** | **Truyền tham chiếu** | | Truyền vào đối tượng các dữ liệu có sẵn(kiểu dữ liệu nguyên thủy). giá trị của hàm thay đổi | Dành cho function, array, giá trị trả về là true và false, giá trị của hàm không thay đổi | |
| 110 | **Câu lệnh return dùng để làm gì?**   * Khi hàm kết thúc quá trình thực thi, nó có thể trả về giá trị cho nơi gọi hàm bằng cách sử dụng từ khóa return. |
| 111 | **Có được phép viết “return;”?**   * Được. * Trả về undefined |
| 112 | **Phân biệt return và break?**   |  |  | | --- | --- | | **Return** | **Break** | | Trả về giá trị của một hàm | Không trả về giá trị mà thoát ra khỏi 1 phương thức | |
| 113 | **Hàm thực hiện việc kiểm tra nên trả về giá trị không? Nếu có, nên trả về gì?**   * Có. * True / false |
| 114 | **Tại sao hàm hoán vị swap(a,b) lại không thực hiện được việc hoán vị của các tham số gốc?**  Ta chỉ có thể thay đổi giá trị đối số truyền vào thông qua tham số gốc |
| 115 | **Trình bày về OOP?**  Lập trình hướng đối tượng OOP (Object Oriented Programming) là một phương pháp lập trình dựa trên các khái niệm về lớp và đối tượng |
| 116 | **Phân biệt giữa lập trình hướng đối tượng và lập trình hướng thủ tục/chức năng?**   |  |  | | --- | --- | | **LT OOP** | **LT hướng thủ tục** | | Hướng tiếp cận đa dạng gần với thực tế  Tìm và phân tích mối quan hệ giữa các đối tượng trong bài toán  Mã lệnh và dữ liệu liên kết theo thể thống nhất  Khả năng tái sử dụng cao ổn định và dễ bảo trì chi phí giảm dần | Hướng tiếp cận từ trên xuống dưới  Chia nhỏ bài toán thành các chức năng  Dữ liệu và mã lệnh phân tán  Tính ổn định giảm khi hệ thống phát triển  Khó bảo trì và tái sử dụng, chi phí hát triển cao | |
| 117 | **Phân biệt giữa object và class?**   * **Lớp(class)** là khuôn mẫu chứa những khai báo định dạng và nguyên tắc hoạt động * **Đối tượng(object)** là sự thể hiện của một lớp và được coi như những sản phẩm thực sự được tạo ra từ khuôn mẫu đó (lớp). |
| 118 | **Hai khái niệm “Môn học” và “Môn học JavaScript” thì đâu là object?**  **Môn học JavaScript** |
| 119 | **Mô tả về sơ đồ lớp?**  Sơ đồ lớp mô tả những đặc điểm khái quát nhất về lớp |
| 120 | **Vẽ sơ đồ lớp của lớp Học viên?**   |  | | --- | | **Student** | | -name: String  -id: int  -age: int | | +say(): void  +move(): void  +write(): void | |
| 121 | **Thuộc tính là gì? Nên dùng từ loại nào để biểu diễn thuộc tính?**   * Thuộc tính: là các thông tin, đặc điểm của đối tượng mà lập trình viên hướng đến. * Đối tượng có các thuộc tính (property) dùng để xác định các đặc tính của nó. * Hành vi là những cách thức mà qua đó đối tượng thể hiện sự hoạt động hay chức năng của chúng |
| 122 | **Phương thức là gì? Nên dùng từ loại nào để biểu diễn phương thức?**  Phương thức: Chính là các thao tác, hành động mà đối tượng lập trình có thể thực hiện được   * Đối tượng còn có các phương thức (method) dùng để định nghĩa cách thức hoạt động của nó. |
| 123 | **Cú pháp định nghĩa 1 class trong JavaScript?** |
| 124 | **Cú pháp tạo 1 đối tượng từ class?** |
| 125 | **Trình bày một số class có sẵn trong JavaScript mà em biết?**   * new Date() * new String() * new Math() * new Array() |
| 126 | **Cú pháp truy cập đến 1 thuộc tính của đối tượng?**   * Dấu chấm (".") để truy xuất các thuộc tính và phương thức của đối tượng |
| 127 | **Cú pháp truy cập đến 1 phương thức của đối tượng?**   * Dấu chấm (".") để truy xuất các thuộc tính và phương thức của đối tượng |
| 128 | **Từ khóa this dùng để làm gì?**   * Từ khóa “this” trả về chính cái đối tượng mà nó thuộc về |
| 129 | Trình bày về DOM (Document Object Model – Mô hình đối tượng tài liệu)?  Là một programming interface cho các tài liệu HTML và XML.  Là một [giao diện lập trình ứng dụng](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng) ([API](https://vi.wikipedia.org/wiki/Giao_di%E1%BB%87n_l%E1%BA%ADp_tr%C3%ACnh_%E1%BB%A9ng_d%E1%BB%A5ng)). Thông thường, DOM có dạng một [cây cấu trúc dữ liệu](https://vi.wikipedia.org/wiki/C%C3%A2y_(c%E1%BA%A5u_tr%C3%BAc_d%E1%BB%AF_li%E1%BB%87u)), được dùng để truy xuất các tài liệu dạng [HTML](https://vi.wikipedia.org/wiki/HTML) và [XML](https://vi.wikipedia.org/wiki/XML). Mô hình DOM độc lập với [hệ điều hành](https://vi.wikipedia.org/wiki/H%E1%BB%87_%C4%91i%E1%BB%81u_h%C3%A0nh) và dựa theo kỹ thuật lập trình hướng đối tượng để mô tả tài liệu. |
| 130 | **Thẻ <canvas> dùng để làm gì?**   * Sử dụng để vẽ các hình ảnh trong trang web. |
| 131 | **Vẽ hình chữ nhật trên canvas?** |
| 132 | **Vẽ hình tròn trên canvas?** |
| 133 | **Vẽ text trên canvas?** |
| 134 | **Nếu muốn di chuyển một hình tròn trên canvas cần làm gì?**  function component(width,height,color,x,y,type){   this.type =type;   if (type== "image"**||type== "background"**){     this.image = new Image();     this.image.src =color;   }   this.width =width;   this.height =height;   this.speedX = 0;   this.speedY = 0;   this.x =x;   this.y =y;   this.update = function(){    ctx=myGameArea.context;     if (type== "image" ||type== "background"){       ctx.drawImage(this.image, this.x, this.y, this.width, this.height); **if (type== "background"){         ctx.drawImage(this.image, this.x + this.width, this.y, this.width, this.height);       }**    } else {       ctx.fillStyle =color;      ctx.fillRect(this.x, this.y, this.width, this.height);     }   }   this.newPos = function(){     this.x += this.speedX;     this.y += this.speedY; **if (this.type == "background"){       if (this.x ==-(this.width)){         this.x = 0;       }     }**  **}**  **}**  } |