**Câu 1:**

|  |  |
| --- | --- |
| Học cách học | * Cornell Notes * Nguyên lí của việc học * Hiểu về não bộ * Kĩ thuật học tập hàng đầu |
| Quản lí công việc bằng Kanban | * Quản lí công việc bằng bảng Kanban * Đặt mục tiêu theo tiêu chuẩn SMART * Thích ứng với khó khăn và thay đổi, đảm bảo chất lượng công việc và liên tục cải tiến |
| Thuật toán | * Thuật toán và các bước giải quyết vấn đề * Hai phương pháp mô tả thuật toán: Pseudo-code & Flowchart |
| Git & HTML | * Tìm hiểu về Git * Một số thao tác cơ bản với Git * Tìm hiểu về một trang web cơ bản * Các thẻ HTML cơ bản và thuộc tính của chúng * Comment trong HTML * HTML Form & Table * Hai phương thức POST và GET trong HTML Form |
| JavaScript | * Nhúng file JS và HTML * Debug trong JS * Một số sự kiện trong JS * Cú pháp JS: Biến, hằng, kiểu dữ liệu, toán tử, … |
| Cấu trúc trong JS | * Cấu trúc rẽ nhánh   + if, if – else, if – else – if,…   + switch – case   + Toán tử ba ngôi * Cấu trúc lặp   + For   + While, Do – while   + Vòng lặp lồng nhau |
| Mảng | * Mảng trong JS * Mảng một chiều, mảng nhiều chiều * Các thao tác với mảng * Các hàm có sẵn của mảng |
| Hàm | * Hàm trong JS * Đối số của hàm * Giá trị trả về của hàm * Gọi hàm * Coding Convention |
| Lập trình hướng đối tượng - OOP | * Tìm hiểu về lớp và đối tượng * Ba cách định nghĩa lớp trong JS * Vẽ trên Canvas * Khai báo và khởi tạo đối tượng |

**Câu 2:**

Giải thuật/ Thuật toán là tập hợp hữu hạn các bước được thực hiện theo một trình tự nào đó để giải quyết vấn đề.

**Câu 3:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Mục đích** | **Cách thực hiện** | **Ưu điểm** | **Nhược điểm** |
| **Pseudo-code** | Dùng để mô tả các bước thực hiện thuật  toán | Sử dụng một tập hợp những từ tương tự như mã thật nhưng nó không thể được biên dịch và thực thi như mã thật | * Đơn giản * Không cần kiến thức về cách biểu diễn | * Dài dòng, không cấu trúc * Đôi lúc khó hiểu, không diễn đạt được thuật toán |
| **Flowchart** | Vẽ ra biểu đồ của luồng chỉ thị hay những hoạt độn,,,g trong một tiến trình. Mỗi hoạt động như vậy được biểu diễn qua những ký hiệu | Quy định chặt chẽ về từng kí hiệu | Cần biết về cách biểu diễn |

**Câu 4:**

Giải thuật giống như là một bộ khung của bài toán. Khi tiếp cận một bài toán, ta nên phân tích thật kĩ bài toán, từ đó đưa ra thuật toán phù hợp để giải quyết bài toán. Sau khi đã có một thuật toán, hãy xem xét lại xem thuật toán đó đã đáp ứng được đủ các yêu cầu của bài toán vừa đặt ra hay chưa hoặc có cách nào khác ngắn gọn hơn để giải quyết bài toán được đưa ra hay không? Sau khi tối ưu thuật toán, lúc này ta đã có một bộ khung thật vững chắc. Lúc này, việc viết code sẽ dễ dàng hơn nhiều, chúng ta chỉ cần viết code dựa theo bộ khung đã có là được.

**Câu 5:**

Với bài toán phức tạp, Flowchart được sử dụng rất hữu ích trong việc truyền đạt. So với Pseudo-code, Flowchart ngoài việc thể hiện được thuật toán, nó còn thể hiện trực quan luồng đi của chương trình, giúp người đọc có thể dễ dàng thực hiện dựa trên đó. Còn Pseudo-code, tuy cũng thể hiện được thuật toán, nhưng nó không thực hiện được luồng đi của chương trình nên ta sẽ khó hình dung ra các bước giải quyết bài toán hơn Flowchart.

**Câu 6:**

Mục tiêu của giải thuật là ta sẽ giải quyết được bài toán, nếu không có hữu hạn các bước thực thi (vô hạn) thì chương trình sẽ không bao giờ kết thúc, đồng nghĩa với việc bài toán sẽ không bao giờ được giải quyết.

**Câu 7:**

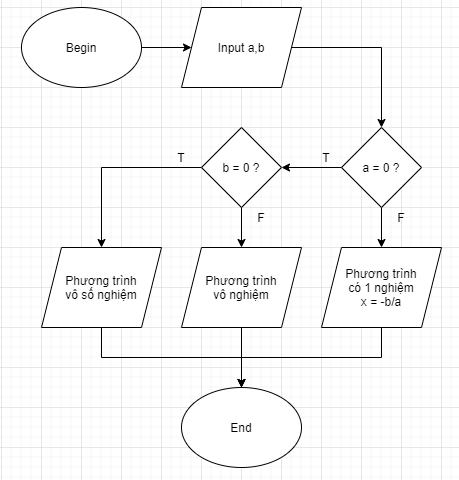
Các bước tính toán

Các lệnh nhập xuất

Cấu trúc rẽ nhánh

Dòng chảy chương trình

**Câu 8:**



**Câu 9:**

VCS viết tắt của Version Control System là một loại phần mềm giúp chúng ta quản lý sự thay đổi trong source code của dự án theo thời gian một cách dễ dàng và thuận tiện.

VCS lưu trữ mọi sự thay đổi trong source code của dự án trong một cơ sở dữ liệu đặc biệt. Chúng ta có thể dễ ràng truy vấn ra được sự thay đổi của source code trong một file tại một thời điểm bất kỳ. Và nếu cần chúng ta có thể dễ dàng đảo ngược source code trong file trở về một thời điểm trước đó.

VCS đặc biệt hữu dụng trong các dự án lớn đòi hỏi sự tham gia của nhiều lập trình viên. Ví dụ tại cùng một thời điểm, chúng ta cần phát triển thêm nhiều feature (tính năng) mới cho phần mềm. Và mỗi thành viên trong team được phân công để phát triển một feature. Khi có hai thành viên cùng hoàn thành xong hai feature riêng biệt cùng lúc thì khi đó việc merge (gộp) code lại có thể sẽ gây ra conflict (xung đột) vì hai người này có thể sửa đổi cùng một đoạn code trên cùng một file.

Lúc này sử dụng VCS chúng ta có thể dễ dàng kiểm tra được sự thay đổi nào đến từ ai và cho feature nào.

**Câu 10:**

Dự theo cách thiết kế thì VCS được chia thành hai kiểu khác nhau đó là VCS tập trung (centralized) và VCS phân tán (distributed).

Với các VCS dạng tập trung thì source code của dự án sẽ được lưu trữ trên một kho tập trụng (hay kho trung tâm) trên một máy chủ. Mỗi lập trình viên muốn tạo ra sự thay đổi cho source code lưu trữ trong kho trung tâm trung (centralized repository) thì họ cần phải thực hiện một công việc trước đó là update source code trên máy tính của họ với kho tập trung trước sau đó mới được thay đổi.

Ngược lại với VCS dạng phân tán thì mỗi lập trình viên sẽ có riêng một kho (repository) của mình và có thể tự do phát triển khác feature và tạo version mới (hay đúng hơn là một commit) mà không cần phải đồng bộ với kho trên server. Điều này sẽ giúp lập trình viên tập trung vào việc hoàn thành feature hay fix bug mà không cần lo về việc phải liên tục update code trên máy tính của họ với code trên server để giảm thiểu xung đột. Việc xử lý xung đột (nếu có) sẽ được thực hiện một lần duy nhất.

**Câu 11:**

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 sẽ giúp bạn:

* Phối hơp thực hiện dự án trong nhóm một cách hiệu quả
* Dễ dàng thử nghiệm, mở rộng tính năng của dự án mà không làm ảnh hưởng đến phiên bản chính (master branch)
* Khôi phục tập tin đã mất
* Phát hiện được ai đã sửa phần nào làm phát sinh lỗi
* Dễ dàng so sánh giữa các phiên bản
* Khôi phục lại mã nguồn từ một phiên bản bất kì
* Lưu lại được các phiên bản khác nhau của mã nguồn dự án phần mềm

**Câu 12:**

Git thuộc hệ thống quản lí mã nguồn phân tán

**Câu 13:**

**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. Chẳng hạn Bob và James mỗi người đều có các Local Repository trên máy của riêng mình. Các đoạn mã do Bob hoặc James viết ra đều được lưu trữ trên các Local Repository đó.

**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. Hoặc chúng ta có thể sử dụng các dịch vụ miễn phí (chẳng hạn như GitHub.com, GitLab.com...). 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.

**Câu 14:**

Đồng bộ giữa các Repository để giúp cho mã nguồn luôn được cập nhật. Làm cho tất cả các Repo là những bản sao của nhau.

**Câu 15:**

Câu lệnh **git push** để đẩy các thay đổi ở Local Repository lên trên Remote Repository.

Cú pháp: git push origin <name\_branch>

**Câu 16:**

Câu lệnh **git pull** sẽ gộp những thay đổi mới kéo về từ máy chủ từ xa với nhánh hiện tại trên máy local.

**Câu 17:**

Câu lệnh **git clone** sẽ tạo một thư mục mới có tên giống trên của repo.

Cú pháp: git clone <link>

**Câu 18:**

**git init** : tạo một thư mục mới có tên .git, thư mục này chứa tất cả các tập tin cần thiết cho kho chứa.

**git remote add origin <remote\_url>** : liên kết Local Repo với Remote Repo

**git add <file\_name>** : cập nhật file lên Staging Area

**git commit –m “message”** : đẩy thông tin thay đổi lên Local Respository với nội dung *message*

**git push –u origin <name\_branch>** : cập nhật thông tin mới commit lên server

**Câu 19:**

**www** viết tắt của cụm từ **World Wide Web** là không gian nơi chứa thông tin, tài liệu và tài nguyên của mọi website trên toàn cầu. **www** được xác định bởi các siêu [liên kết URL](https://wiki.matbao.net/kb/url-la-gi-friendly-url-la-gi-huong-dan-toi-uu-url-cho-seo/) và truy cập thông qua Internet.

**Câu 20:**

**HTML** viết tắt của **Hypertext Markup Language** là ngôn ngữ lập trình dùng để xây dựng và cấu trúc lại các thành phần có trong website. HTML tạm dịch là ngôn ngữ đánh dấu siêu văn bản.

**Câu 21:**

Thẻ tiêu đề: h1, h2,…, h6

Thẻ tạo văn bản: p

Thẻ tạo danh sách: ul, ol, li

**Câu 22:**

Thẻ **<div>**, div là viết tắt của từ division ( khu ) được dùng để tạo một khu vực nào đó trên website.

**<div>** có kiểu hiển thị là block, trong khi **<span>** là một thẻ có kiểu hiện thị là inline

**Câu 23:**

**URL** là từ viết tắt của cụm từ tiếng Anh: “**Uniform Resource Locator**” có nghĩa tiếng Việt là “Tham chiếu tài nguyên internet”. Nó là một phương tiện để người dùng sử dụng, truy cập đến tài nguyên trên mạng máy tính. Mỗi tài nguyên trên máy tính khi ta truy cập tìm kiếm hàng ngày đều được gán một địa chỉ, hay còn gọi là đường link cố định (thay vì một dãy địa chỉ IP dài). Chỉ cần nhập địa chỉ (đường link) đó và search, bạn có thể đến được địa chỉ website mình tìm kiếm.

**URL tuyệt đối** (là một chuỗi đầy đủ bao gổm *http://*, *tên miền của trang web*, *đường dẫn đến tập tin*)

**URL tương đối** là một phần nhỏ của đường dẫn tuyệt đối (thông thường đường dẫn tương đối là phần *đường dẫn đến tập tin*)

Ví dụ:

**URL tuyệt đối:** http://google.com.vn/html/default.html

**URL tương đối:** html/default.html

Câu 24:

**Hyperlink** có nghĩa là một **siêu**[**liên kết**](https://seoantoan.com/backlink-toan-tap-cach-tao-backlink-chat-luong-check-backlink), là nguồn dẫn tài liệu để khi người dùng click vào nó sẽ được dẫn đến một trang khác hoặc một vị trí nào đó trên trang web.

Câu 25:

Sử dụng thẻ <a>, trong thuộc tính **href** thì chèn ‘#’ và id của phần mà bạn muốn đi tới trên trang web

VD: <p id=head>Link to here</p>

<a href = ‘#head’> Click here to go to header</a>

Câu 26:

Sử dụng thẻ <a>, trong thuộc tính **href** thì chèn url của trang web mà bạn muốn tới

VD: <a href = ‘https://google.com.vn’> Click here to go to Google</a>

Câu 27:

**Internal Link** là một liên kết nội bộ từ trang này sang trang khác trên cùng một tên miền (hay website). Chúng thường được sử dụng trong việc điều hướng và chia sẻ giá trị liên kết giúp cải thiện khả năng xếp hạng của website.

Liên kết ngoài (External link hay [Backlink](https://seothetop.com/xay-dung-link/xay-dung-backlink-mu-trang-157106.html)) là liên kết từ một trang khác đến trang của bạn.

Câu 28:

Trong HTML có ba kiểu danh sách (list type) đó là kiểu sắp xếp (ordered list), kiểu không sắp xếp (unordered list) và kiểu danh sách mô tả (description list). Cụ thể:

* Kiểu sắp xếp (**Ordered List**): Là kiểu hiển thị một danh sách mà các mục con của nó được sắp xếp theo thứ tự bằng số hoặc chữ cái.

<ol> //Cặp thẻ bắt đầu

<li> </li> //Các mục con bên trong

</ol>

* Kiểu không sắp xếp (**Unordered List**): Là kiểu hiển thị danh sách mà các mục con của nó sẽ không được sắp xếp theo thứ tự mà chỉ được đánh dấu bằng một ký tự đặc trưng.

<ul> //Cặp thẻ bắt đầu

<li> </li> //Các mục con bên trong

</ul>

* Kiểu mô tả (**Description List**): Là kiểu hiển thị danh sách mà các mục con của nó sẽ không được đánh dấu thứ tự, nhưng sẽ có kèm theo một đoạn miêu tả.

<dl> //Cặp thẻ bắt đầu

<dt> </dt> //Tên mục con

<dd> </dd> //Mô tả cho mục con

</dl>

Câu 29:

Dùng thẻ <ol> với thuộc tính type=‘a’ và start=‘3’.

VD: <ol type="a" start="3">  
 <li>Coffee</li>  
 <li>Tea</li>  
 <li>Coca-Cola</li>  
</ol>

Câu 30:

Form hay biểu mẫu dùng để thu thập thông tin từ phía người dùng.

Câu 31:

Thuộc tính action dùng để định nghĩa hành động sẽ được thực hiện khi nhấn nút gửi dữ liệu, hành động này có thể là gửi dữ liệu lên web server, gửi dữ liệu đến một mail server.

Câu 32:

Mặc định thuộc tính method của form có giá trị là GET. Khi sử dụng phương thức GET, dữ liệu của form sẽ được đưa vào kèm theo với đường dẫn URL gửi đến máy chủ web.

Khi sử dụng phương thức POST với form nhập liệu, các dữ liệu form sẽ không được hiển thị trong URL mà nó sẽ đóng gói trong header của gói tin gửi đi, do đó các thông tin sẽ được bảo mật hơn. Với phương thức POST, độ lớn của dữ liệu gửi đi là không giới hạn, tuy nhiên gửi dữ liệu kiểu POST không thể đánh dấu để sử dụng lại sau.

Câu 33:

<input type= ‘**button**’ />

//Sử dụng như một button, không có gì xảy ra khi click.

<input type= ‘**submit**’ />

//Khi nhấn sẽ gửi dữ liệu lên server thông qua thuộc tính action và phương thức của method

<input type= ‘**reset**’ />

//Khi nhấn sẽ reset toàn bộ giá trị về mặc định

Câu 34:

<form action="CodeGym.vn" method="get">  
 <input type="submit">  
</form>

Câu 35:

Sử dụng thuộc tính **colspan=‘3’** trong thẻ <td> hoặc <th>. VD:

<td colspan="3"></td>

Câu 36:

Sử dụng thuộc tính **rowspan=‘2’** trong thẻ <td> hoặc <th>. VD:

<th colspan="2"></th>

Câu 37:

**Thuộc tính cellspacing**dùng để tạo khoảng cách giữa các ô trong table

**Thuộc tính cellpadding** dùng để tạo khoảng cách giữa nội dung trong ô so với đường viền

Câu 38:

Có một số cách khác nhau để nhúng mã Javascript vào một trang web:

* Sử dụng thẻ **<script**> bên trong mã HTML
* Sử dụng file Javascript riêng biệt
* Viết mã Javascript ngay trong thẻ các thẻ html

Câu 40:

JavaScript **có** phân biệt chữ hoa chữ thường. VD: A và a là hai biến khác nhau

Câu 41:

**camelCase**: Chữ cái đầu tiên của từ đầu tiên viết thường. Các từ còn lại viết hoa chữ cái đầu tiên. Thường sử dụng để đặt tên biến, hàm, phương thức

VD: createCircle, getArea, inputNumber,…

**CamelCase**: Quy tắc này yêu cầu viết hoa các chữ cái đầu tiên của mọi từ. Thường sử dụng để đặt tên class.

VD: PuzzleGame, FlashLamp,….

Câu 42:

**Biến** là tên của vùng nhớ dùng để lưu trữ dữ liệu.

**Biến cục bộ (local)**: Một biến được gọi là cục bộ khi bạn khai báo nó nằm bên trong một hàm cụ thể nào đó, lúc này biến đó sẽ không sử dụng được ở bên ngoài [hàm](https://freetuts.net/ham-va-tao-ham-function-trong-javascript-274.html). (Sử dụng từ khoá let; khai báo trong hàm hoặc trong khoảng dấu {} )

**Biến toàn cục (global)**: là biến mà bạn khai báo bên ngoài và không nằm bên trong một hàm cụ thể nào cả. (Sử dụng từ khoá let hoặc var; khi khai báo trong hàm hoặc khoảng dấu {} mà dùng var thì biến đó vẫn là biến toàn cục)

Câu 43:

**const** được sử dụng để khai báo 1 hằng số, và giá trị của nó không thay đổi trong suốt chương trình.

**let** khai báo biến chỉ có thể truy cập được trong block bao quanh nó được xác định bằng cặp {}.

**var** khai báo biến có thể truy cập ở phạm vi hàm số hoặc bên ngoài hàm số, toàn cục.

Câu 44:

JavaScript là một ngôn ngữ định kiểu yếu hay 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.

VD: let foo = 42; // foo là một số

foo = 'bar'; // foo bây giờ là một chuỗi

foo = true; // foo bây giờ là một boolean

Câu 45:

Trong JS có 5 kiểu dữ liệu nguyên thuỷ:

* Number
* Boolean
* String
* Null
* Undefined

Câu 46:

Toán tử **typeof** trả về kiểu dữ liệu của đối tượng nào đó.

VD: let x=10; // x có kiểu dữ liệu là number (typeof(x) = ‘number’)

Câu 47:

Có 3 cách đưa thông báo trong JS:

* alert();

// dùng để hiển thị một hộp thoại thông báo đến người dùng

* confirm();

// dùng để hiển thị một hộp thoại thông báo và yêu cầu người dùng phải xác nhận là đồng ý hoặc quay lại (OK hoặc Cancel)

* prompt();

// dùng để hiển thị một hộp thoại thông báo và yêu cầu người dùng nhập dữ liệu vào

Câu 48:

* Toán tử một ngôi: ++, --, - //++ & -- dùng để tăng hoặc giảm đi 1 đơn vị, - thể hiện việc thao tác với số đối của nó (a = 5; -a = -5)
* Toán tử gán: =, +=, -= , \*=, /= , %= // gán giá trị cho biến
* Toán tử toán học: +,-,\*,/,% // thực hiện xử lý thay đổi giá trị các biến
* Toán tử quan hệ: <,<=,>,>=,==,===,!==,!=== // thể hiện mối quan hệ giữa hai toán tử trong toán học
* Toán tử luận lí: && , || , ! // dùng để kết hợp hay phủ định biểu thức có chứa các toán tử quan hệ
* Toán tử ba ngôi: (a>b) ? max = a : max = b // if(a>b) max = a; else max = b;

Câu 49:

== : So sánh hai giá trị với nhau

=== : So sánh hai giá trị và hai kiểu dữ liệu với nhau

Câu 50:

&& : Trả về giá trị TRUE khi tất cả các toán hạng đều TRUE và FALSE khi ít nhất một toán hạng FALSE

|| : Trả về giá trị TRUE khi ít nhất một toán hạng TRUE và FALSE khi tất cả các toán hạng FALSE

Câu 51:

+0===-0 // trả về TRUE vì số đối của 0 là 0 và cùng là kiểu dữ liệu number.

Câu 52 + 53:

* ++ : Dùng để tăng giá trị thêm 1 đơn vị
  + a++ : thực hiện biểu thức trước sau đó mới tăng giá trị của a
  + ++a : tăng giá trị của a trước rồi mới thực hiện biểu thức
* -- : Dùng để giảm giá trị đi 1 đơn vị
  + a-- : thực hiện biểu thức trước sau đó mới giảm giá trị của a
  + --a : giảm giá trị của a trước rồi mới thực hiện biểu thức
* - : Dùng để thao tác với số đối //a=5; (-a = -5)

Câu 54:

Câu lệnh điều kiện được sử dụng khi bạn muốn thực hiện các hành động khác nhau đối với các quyết định (điều kiện) khác nhau.

Câu 55:

Trong JavaScript, chúng ta có những lệnh điều kiện sau đây:

* Sử dụng **if**để xác định một khối mã được thực thi, nếu một biểu thức điều kiện được đánh giá là true
* Sử dụng **else** để xác định một khối mã được thực thi, nếu điều kiện được chỉ ra trong **if** là sai
* Sử dụng **else if** nhằm xác định một điều kiện mới để kiểm tra, nếu các điều kiện trước đó là sai
* Sử dụng **switch** để xác định nhiều khối mã luân phiên được thực thi

Câu 56:

Sử dụng lệnh **if** để xác định một khối mã JavaScript sẽ được thực thi nếu điều kiện chỉ ra là đúng.

if (biểu thức điều kiện) {  
    khối mã được thực thi nếu điều kiện là đúng  
}

Sử dụng phẩn mở rộng **else** để xác định một khối mã được thực thi nếu điều kiện là sai.

if (biểu thức điều kiện) {  
    khối mã được thực thi nếu điều kiện là đúng  
} else {  
    khối mã được thực thi nếu điều kiện trên là sai  
}

Câu 57:

Lệnh **switch** được sử dụng để thực hiện các hành động khác nhau dựa trên các điều kiện khác nhau.

switch(biểu thức) {  
  case giá trị 1:  
    // khối lệnh sẽ được thực thi  
    break;  
  case giá trị 2:  
    // khối lệnh sẽ được thực thi  
    break;  
  default:  
    // khối lệnh mặc định sẽ được thực thi nếu không chọn được trường hợp nào  
  }

Câu 58:

Else là một tuỳ chọn của câu lệnh if vì nếu thiếu else thì chương trình vẫn hoạt động bình thường mà không hề phát sinh lỗi.

Câu 59:

Nên sử dụng câu lệnh switch khi biểu thức điều kiện bắt buộc là so sánh bằng và số điều kiện nhiều. Việc sử dụng switch – case sẽ giúp cấu trúc chương trình rõ ràng, dễ đọc, dễ hiểu hơn.

Câu 60:

Khi trình thông dịch (interpreter) mã JavaScript gặp từ khóa **break**, nó thoát ra khỏi khối switch hoặc vòng lặp gần nhất.

Điều này sẽ làm ngừng thực thi thêm các mã và các case bên trong khối lệnh.

Nếu không có câu lệnh break, chương trình sẽ thực thi từ case đúng cho đến hết khối lệnh theo tuần tự từ trên xuống dưới.

Câu 61:

Mệnh đề default **không** bắt buộc trong câu lệnh switch.

Câu 62:

Toán tử 3 ngôi là cách viết khác của câu lệnh điều kiện if-else. Cú pháp:

(condition) ? value1 : value2

Nếu condition nó là đúng (hay là true) thì toán tử sẽ trả về value 1 còn nếu là sai (hay là false) thì nó sẽ trả về giá trị của value 2

Câu 63:

Event là sự tương tác của JavaScript với HMTL được xử lý thông qua các sự kiện xảy ra khi người sử dụng hoặc trình duyệt thao tác một trang

Câu 64:

Sự kiện onchange: Xảy ra khi giá trị (value) của thẻ HTML đổi. Thường dùng trong các đối thẻ form input

Câu 65:

Sự kiện onclick: Xảy ra khi click vào thẻ HTML

Câu 66:

<button onclick="myFunction()">Delete</button>

<script>

function myFunction() {  
 confirm("Bạn có chắc chắn muốn xoá không?");  
 }  
</script>

Câu 67:

**onmouseover :** Xảy ra khi con trỏ chuột bắt đầu đi vào thẻ HTML

**onmouseout :** Xảy ra khi con trỏ chuột bắt đầu rời khỏi thẻ HTML

**onmousemove:** Xảy ra khi con chuột di chuyển bên trong thẻ HTML

Câu 68:

**onkeyup :** Xảy ra khi bạn gõ phím nhưng lúc bạn nhả phím ra sẽ được kích hoạt

**onkeydown :** Xảy ra khi gõ một phím bất kì vào ô input

**onkeypress :** Xảy ra khi bạn nhấn một phím vào ô input

Câu 69:

**onfocus:** Được kích hoạt khi người sử dụng đặt focus vào một cửa sổ, khung hay phần tử form.

**onblur :** Khi người dùng rời khỏi một phần tử trên form, sự kiện onBlur được kích hoạt.

Câu 70:

**onload :** Xảy ra khi thẻ HTML bắt đầu chạy

Câu 71:

**innerHTML**: chèn một đoạn mã HTML vào trong thẻ mà mình muốn chèn

**innerText**: chèn một đoạn text đơn thuần vào trong thẻ mà mình muốn chèn

Câu 72:

Thêm thuộc tính **disabled** vào thẻ mà bạn muốn vô hiệu hoá

Câu 73:

**readonly**quy định giá trị trong thẻ input chỉ được dùng để hiển thị (không thể thay đổi). Khi thuộc tính này được thiết lập, bạn vẫn có thể sử dụng thẻ.

**disabled** được sử dụng để vô hiệu hóa một thẻ input. Khi thuộc tính này được thiết lập, bạn không thể nhấn hay sử dụng thẻ.

Câu 74:

**Vòng lặp** được sử dụng để thực thi một đoạn code **lặp** đi **lặp** lại một số lần nhất định dựa trên giá trị trả về từ một điều kiện cho trước.

Câu 75:

Có 3 loại vòng lặp cơ bản trong JS:

* **for** - lặp qua một khối lệnh một số lần
* **while** - lặp qua một khối lệnh trong khi biểu thức điều kiện đúng
* **do-while** - Cũng là lặp qua một khối lệnh trong khi biểu thức điều kiện đúng

Câu 76:

For dùng trong vòng lặp với số lần biết trước và kiểm soát được.

While dùng trong vòng lặp với số lần ko biết trước, kiểm tra biểu thức điều kiện trước rồi mới thực hiện câu lệnh

Do-while dùng trong vòng lặp với số lần ko biết trước, nó thực hiện câu lệnh trước rồi mới kiểm tra biểu thức điều kiện

Câu 77:

Sử dụng **do-while** khi cần làm công việc rồi mới kiểm tra điều kiện, ví dụ như kiểm tra các giá trị khi người dùng nhập dữ liệu,…

Câu 78:

Sử dụng **for** khi có thể xác định trước số lần cần lặp.

Câu 79:

Sử dụng **while** trong trường hợp không xác định được rõ số lần lặp

Câu 80:

**Có** nên sử dụng vòng lặp, sử dụng **do-while** để thực hiện việc kiểm tra

Câu 81:

Nên dùng vòng lặp **for** để thực hiện

Câu 82:

Vòng lặp for cho phép bỏ qua một hoặc tất cả các thành phần nếu không sử dụng chúng.

Ví dụ:

for ( ; ; )  
{  
    statements;  
}

Câu 83:

for(let i=1,j=9;i<=j;i++,j--){  
 console.log(i+" "+j);  
}

Câu 84:

for(let i=0;i<5;i++){  
 document.write("<br>");  
 for(let j=0;j<=i;j++){  
 document.write("\* ");  
 }  
}

Câu 85:

Giả sử vòng lặp for bên ngoài lặp m lần, vòng for trong lặp n lần. Các câu lệnh bên trong vòng for thứ hai sẽ được thực hiện m\*n lần.

Câu 86:

Khi gặp câu lệnh **break**, chương trình sẽ thoát khỏi vòng for và thực hiện tiếp khối lệnh tiếp theo.

Khi gặp câu lệnh **continue**, chương trình sẽ tạm dừng tất cả câu lệnh sau continue và tiếp tục thực hiện vòng for kế tiếp.

Câu 87:

for(let i=1;i<100;i++){  
 if(i%5===0){  
 continue;  
 }  
 console.log(i);  
}

Câu 88:

Mảng là một tập hợp các phần tử lại và mỗi phần tử sẽ được đánh dấu một vị trí trong tập hợp đó.

Câu 89:

Có hai cách thông thường để khai báo mảng

* Sử dụng từ khóa new Array() // let name\_array = new Array(1,2,3);
* Sử dụng cặp dấu ngoặc vuông ([]). // let name\_array = [1,2,3];

Câu 90:

Chỉ số của mảng là chỉ số đánh dấu vị trí các phần tử trong mảng. Chỉ số của mảng bắt đầu từ 0

Câu 91:

Kích thước của mảng dùng để biểu thị số phần tử hiện có của mảng.

Câu 92:

**push()**  : Thêm phần tử vào cuối mảng

**unshift()** : Thêm phần tử vào đầu mảng

Câu 93:

**push()**  : Thêm phần tử vào cuối mảng

**pop()** : Xoá phần tử cuối mảng

Câu 94:

**shift()**  : Xoá phần tử đầu mảng

**unshift()** : Thêm phần tử vào đầu mảng

Câu 95:

Muốn xoá phần tử cuối cùng của mảng, ta dùng hàm **pop()**

Câu 96:

Muốn xoá phần tử đầu tiên của mảng, ta dùng hàm **shift()**

Câu 97:

Muốn sắp xếp mảng, ta dùng hàm **sort()**

Câu 98:

let arr=["đạt",'thành','phương','việt'];  
arr.sort();  
Câu 99:

Duyệt mảng là duyệt qua từng phần tử trong mảng.

let name\_array = [1,2,3];

for (let i = 0; i < name\_array.length; i++){

    document.write(name\_array[i]);

}

Câu 100:

let name\_array = [[1, 2], [3, 4], [5, 6]];  
for (let i = 0; i < name\_array.length; i++) {  
 for (let j = 0; j < name\_array[i].length; j++) {  
 console.log(name\_array[i][j]);  
 }  
}

Câu 101:

Mảng 2 chiều cần 2 chỉ số để xác định duy nhất một phần tử.

Câu 102:

Tạo một biến sum và gán giá trị ban đầu là 0 để lưu tổng của các phần tử của ma trận số

Sử dụng 2 vòng lặp for lồng nhau. Vòng for thứ nhất lặp qua tất cả các hàng. Vòng for thứ hai lặp qua từng phần tử trong một hàng. Trong vòng for thứ hai thì qua mỗi lần lặp cho biến sum cộng thêm giá trị bằng giá trị của phần tử đang trỏ tới. Sau khi chạy xong hai vòng for thì hiện thị ra giá trị của biến sum chính là tổng của tất cả phần tử của ma trận số.

Câu 103:

Sử dụng 2 vòng lặp for lồng nhau. Vòng for thứ nhất lặp qua tất cả các hàng. Tạo một biến sum và gán giá trị ban đầu là 0 để lưu tổng của các phần tử trong một hàng. Vòng for thứ hai lặp qua từng phần tử trong một hàng. Trong vòng for thứ hai thì qua mỗi lần lặp cho biến sum cộng thêm giá trị bằng giá trị của phần tử đang trỏ tới. Sau khi chạy xong hai vòng for thì hiện thị ra giá trị của biến sum chính là tổng của tất cả phần tử của hàng đó.

Câu 104:

Sử dụng 2 vòng lặp for lồng nhau. Vòng for thứ nhất lặp qua tất cả các cột. Tạo một biến sum và gán giá trị ban đầu là 0 để lưu tổng của các phần tử trong một cột. Vòng for thứ hai lặp qua từng phần tử trong một cột. Trong vòng for thứ hai thì qua mỗi lần lặp cho biến sum cộng thêm giá trị bằng giá trị của phần tử đang trỏ tới. Sau khi chạy xong hai vòng for thì hiện thị ra giá trị của biến sum chính là tổng của tất cả phần tử của cột đó.

Câu 105:

Hàm sẽ gom một số đoạn code vào một khối xử lý và khi cần thì gọi ra dùng.

Câu 106:

Tham số hình thức là biến được định nghĩa trong hàm.

Tham số thực là giá trị thực tế truyền vào cho tham số hình thức khi gọi hàm.

Câu 107:

Gọi hàm (invoke) là câu lệnh dùng để thực thi một khối lệnh nào đó đã được lập trình viên định nghĩa.

Câu 108:

**Calling Function**: Được lập trình viên trực tiếp gọi ra khi cần sử dụng với tham số truyền vào là tham số thực

**Called Function**: Là hàm định nghĩa các câu lệnh sẽ được thực hiện, tham số truyền vào là tham số hình thức

Câu 109:

* Giống nhau: đều truyền giá trị vào hàm khi hàm được gọi
* Khác nhau:
  + **Truyền tham trị**: truyền **giá trị** của biến vào hàm.
  + **Truyền tham chiếu**: truyền **địa chỉ** của biến vào hàm.

Câu 110:

Câu lệnh **return** sẽ khiến sự thực thi dừng chương trình con hiện tại và trở về lại lập tức vào thời điểm trong mã sau khi chương trình con đó được gọi.

Câu 111:

Được phép viết **“return;”** .

Câu 112:

**break**: dùng để thoát khỏi vòng lặp, hoặc câu lệnh switch case.

**return**: dùng để thoát ra gọi hàm hoặc phương thức, có thể trả về một giá trị nào đó cho lời gọi hàm.

Câu 113:

Có. Nên trả về giá trị TRUE khi việc kiểm tra điều kiện là đúng và FALSE nếu ngược lại.

Câu 114:

Vì khi gọi tới hàm swap(a,b), các tham số thực sẽ được sao chép vào các tham số hình thức. Việc hoán đổi chỉ đổi giá trị các tham số hình thức và khi kết thúc hàm, toàn bộ các tham số đó sẽ bị xoá đi. Vì vậy, các tham số thực sẽ không bị ảnh hưởng nên hàm swap(a.b) sẽ không thể thực hiện hoán vị với các tham số gốc.

Câu 115:

Lập trình hương đối tượng:

* Không tập trung vào cấu trúc, tập trung vào mô hình hoá dữ liệu
* Sử dụng các lớp khi lập trình
* Mô phỏng các đối tượng trong thế giới thực vào trong các chương trình
* OOP (Object Oriented Programming) là một triết lý xây dựng chương trình
* Tất cả mọi thứ trong OOP đều là đối tượng. Một chương trình được coi như là một thế giới bao gồm các đối tượng
* Mã lệnh và dữ liệu được kết hợp trong một thể thống nhất – là đối tượng
* Đối tượng bao gồm:
  + Thuộc tính: các dữ liệu, tính chất của đối tượng
  + Hành vi: các khả năng, hành động mà đối tượng có thể thực hiện
* Các đối tượng có thể có quan hệ với nhau

Câu 116:

* Lập trình thủ tục:
* Hướng tiếp cận từ trên xuống
* Chia nhỏ bài toán thành các module chức năng
* Dữ liệu và mã lệnh phân tán
* Hạn chế
* Tính ổn định giảm khi hệ thống phát triển
* Khó bảo trì và tái sử dụng
* Chi phí phát triển cao
* OOP:
* 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 trong thể thống nhất
* Ưu thế:
* Khả năng tái sử dụng cao
* Ổn định và dễ bảo trì
* Chi phí giảm dần

Câu 117:

**Class** và **Object** (**Lớp** và **Đối tượng**) là hai trong những khái niệm quan trọng nhất của ngôn ngữ lập trình hướng đối tượng (OOP).

Sự khác biệt chính giữa một Class và một Object trong JS là:

Class là một mô hình chi tiết để bạn sử dụng tạo ra các Object. Class định nghĩa tất cả các thuộc tính và các phương thức cần thiết của một Object.

Mỗi Object phải thuộc một Class nào đó. Và một Object là một thể hiện của Class. Tất cả các Object thuộc về cùng một Class có cùng các thuộc tính và các phương thức.

Câu 118:

‘Môn học’ là Class. ‘Môn học JavaScript’ là object thuộc class ‘Môn học’

Câu 119:

Class Diagram là một loại sơ đồ cấu trúc tĩnh mô tả cấu trúc của hệ thống bằng cách hiển thị các lớp của hệ thống, thuộc tính, hoạt động của chúng và mối quan hệ giữa các đối tượng. Sơ đồ lớp là khối xây dựng chính của mô hình hướng đối tượng.

Class là thành phần chính của bản vẽ Class Diagram. Class mô tả về một nhóm đối tượng có cùng tính chất, hành động trong hệ thống. Class được mô tả gồm tên Class, thuộc tính và phương thức.

|  |
| --- |
| Class Name |
| Attributes |
| Methods |

Class Name: là tên của lớp.

Attributes (thuộc tính): mô tả tính chất của các đối tượng.

Method (Phương thức): chỉ các hành động mà đối tượng này có thể thực hiện trong hệ thống. Nó thể hiện hành vi của các đối tượng do lớp này tạo ra.

Câu 120:

|  |
| --- |
| Học viên |
| * name: string * age: number * class: string * gender: boolean |
| * readAnything() * writeAnything() * speakAnything() |

Câu 121:

Attributes (thuộc tính): mô tả tính chất của các đối tượng. Nên dùng danh từ để biểu diễn thuộc tính

Câu 122:

Method (Phương thức): chỉ các hành động mà đối tượng này có thể thực hiện trong hệ thống. Nó thể hiện hành vi của các đối tượng do lớp này tạo ra. Phương thức thường bắt đầu bằng một động từ

Câu 123:

Các cách định nghĩa class trong JS:

* Sử dụng hàm: Bạn định nghĩa một hàm, sau đó tạo đối tượng sử dụng từ khóa **new**. Để định nghĩa các thuộc tính và phương thức cho một đối tượng sử dụng từ khóa function() và this để truy xuất. VD:

function Apple (type) {

    this.type = type;

    this.color = "red";

    this.getInfo = function() {

        return this.color + ' ' + this.type + ' apple';

    };

}

* Sử dụng object literals: Literals là một cách tạo đối tượng và mảng trong Javascript ngắn nhất trong Javascript. Để tạo ra một đối tượng rỗng. VD:

let apple = {

    type: "macintosh",

    color: "red",

    getInfo: function () {

        return this.color + ' ' + this.type + ' apple';

    }

}

* Singleton sử dụng một hàm: Bạn có thể sử dụng một hàm để xác định một đối tượng singleton

let apple = new function() {

    this.type = "macintosh";

    this.color = "red";

    this.getInfo = function () {

        return this.color + ' ' + this.type + ' apple';

    };

}

Câu 126:

Có 2 cách để truy cập đến thuộc tính đối tượng để lấy giá trị thuộc tính hoặc gán giá trị vào thuộc tính, ví dụ:

objectName.propertyName //Cách 1

objectName['propertyName'] //Cách 2

Câu 127:

Cú phá truy cập đến phương thức của đối tượng, ví dụ:

objectName.methodName()

Câu 128:

Từ khóa **this** trong JS là một biến tham chiếu được sử dụng để tham chiếu tới đối tượng của lớp hiện tại.

Câu 129:

DOM (mô hình đối tượng tài liệu) là một programming interface cho các tài liệu HTML và XML. Nó đại diện cho trang để các chương trình có thể thay đổi cấu trúc document, style và nội dung. DOM mô tả document dưới dạng các nodes và objects. Bằng cách đó, ngôn ngữ lập trình có thể kết nối với trang.

Một trang web là một document. Document này có thể được hiển thị trong cửa sổ trình duyệt, hoặc dưới dạng mã nguồn HTML. DOM đại diện cho document để nó có thể được thao tác bằng các ngôn ngữ lập trình. DOM là một object-oriented đại diện của trang web, có thể được sửa đổi bằng một ngôn ngữ kịch bản như JavaScript.

Câu 130:

**Canvas** là một phần tử của HTML5, cho phép thực hiện lập trình kết xuất đồ họa các đối tượng hai chiều trên trang web.

Câu 131:

Vẽ hình chữ nhật trên Canvas

let c = document.getElementById("myCanvas");

let ctx = c.getContext("2d");

ctx.fillStyle = "#FF0000";

ctx.fillRect(0,0,150,75);

Câu 132:

Vẽ hình tròn trên Canvas:

let c = document.getElementById("myCanvas");

let ctx = c.getContext("2d");

ctx.beginPath();

ctx.arc(95,50,40,0,2\*Math.PI);

ctx.stroke();

Câu 133:

Vẽ chữ trên Canvas

let c = document.getElementById("myCanvas");

let ctx = c.getContext("2d");

ctx.font = "30px Arial";

ctx.fillText("Hello World",10,50);

Câu 134:

// ví dụ về di chuyển hình tròn

let c = document.getElementById('myCanvas');  
c.width = 1000;  
c.height = 1000;  
let ctx = c.getContext('2d');  
let x=50;  
let y=50;  
function *createCircle*(*x*,*y*){  
 ctx.beginPath();  
 ctx.arc(*x*,*y*,30,0,2\*Math.PI);  
 ctx.fillStyle='black';  
 ctx.fill();  
}  
function *clearScreen*(){  
 ctx.fillStyle = 'white';  
 ctx.fillRect(0,0,1000,1000)  
}  
*setInterval*(()=>{  
 *clearScreen*();  
 x+=10;  
 y+=10;  
 *createCircle*(x,y);  
},500)