**BỘ CÂU HỎI PHỎNG VẤN BP 2.0**

## Câu 1: Hệ thống các kiến thức đã học ở Module 1

|  |  |
| --- | --- |
| 1. Học cách học | * Ghi chép hiệu quả bằng Cornell Notes * Ngộ nhận về cách học * Hiểu về não bộ * Dùng Pomodoro để tăng năng suất làm việc * Nguyên lý của việc học * Các kĩ thuật học tập hàng đầu   + Sức mạnh của thái độ   + Các câu hỏi Socrates   + Hướng dẫn viết Phản tư (Reflection) * Đọc sách hiệu quả |
| 2. Hoàn thành mọi việc với Kanban | * Tạo lập thói quen Scrum Life để làm việc vui vẻ và hiệu quả * Đặt mục tiêu tốt cho ngắn hạn và dài hạn * Lên kế hoạch linh hoạt và quản lý công việc với bảng Kanban * Giới hạn công việc đang làm để tập trung hoàn thành từng việc một * Sớm nhận biết các khó khăn và thay đổi nhằm thích ứng và giữ cho công việc luôn trôi chảy * Nâng cao chất lượng công việc dựa trên Định nghĩa Hoàn thành * Cải tiến liên tục để tiến bộ không ngừng |
| 3. Thuật toán/Giải thuật | * Thuật toán và các bước giải quyết vấn đề * Hai phương pháp mô tả thuật toán: Pseudo-code & Flowchart |
| 4. 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 |
| 5. JavaScript Overview | * Nhúng JavaScript vào trang web * Debug trong JS * Một số sự kiện trong JS * Cú pháp JS: Biến, hằng, kiểu dữ liệu, toán tử, … |
| 6. Cấu trúc trong JS | * Cấu trúc rẽ nhánh (Conditions)   + if, if – else, if – else – if,…   + switch – case   + Toán tử ba ngôi * Cấu trúc lặp (Loops)   + For   + While, Do – while   + Vòng lặp lồng nhau |
| 7. Mảng (Arrays) | * Mảng trong JS * Mảng một chiều, mảng nhiều chiều * Các thao tác với mảng * Các hàm có sẵn của mảng |
| 8. Hàm (Functions) | * Hàm trong JS * Đối số của hàm * Giá trị trả về của hàm * Gọi hàm * Coding Convention |
| 9. 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 gì?

là một bộ các quy tắc hay quy trình cụ thể nhằm giải quyết một vấn đề nào đó trong một số bước hữu hạn, hoặc nhằm cung cấp một kết quả từ một tập hợp của các dữ kiện đưa vào.

## Câu 3: So sánh việc trình bày giải thuật bằng pseudo-code và flowchart?

|  |  |  |
| --- | --- | --- |
| **Cách thực hiện** | **Ưu điểm** | **Nhược điểm** |
| Mã giả (Pseudo-code) sử dụng ngôn ngữ tự nhiên để biểu diễn thuật toán 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 |
| Lưu đồ (Flowchart) Một lưu đồ là một hình ảnh minh họa cho giải thuật.  Lưu đồ mô tả giải thuật bằng các sơ đồ hình khối, mỗi khối quy định một hành động. | Quy định chặt chẽ về từng kí hiệu | Cần biết về cách biểu diễn |

## Câu 4: Tại sao lại cần trình bày giải thuật trước khi bắt tay vào code?

Tại vì giải thuật giúp chúng ta phân tích, xác định rõ ràng các bước để giải quyết vấn đề, tìm ra giải pháp tối ưu nhất, ít tốn thời gian và công sức nhất trước khi bắt tay vào code.

## Câu 5: Tại sao lại nên sử dụng Flowchart hơn Pseudo-code khi thể hiện giải thuật của bài toán phức tạp?

Với bài toán phức tạp, sử dụng Flowchart rất hữu ích trong việc truyền đạt. 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.

Với 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: Tại sao giải thuật lại cần hữu hạn các bước thực thi?

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: Trình bày ý nghĩa các ký hiệu sử dụng khi vẽ flowchart?

* Ký hiệu hình eclipse: Bắt đầu hay kết thúc lưu đồ
* Ký hiệu hình bình hành: Các lệnh xuất/nhập
* Ký hiệu hình chữ nhật: Các bước tính toán, xử lý
* Ký hiệu hình thoi: Các bước cần rẽ nhánh
* Ký hiệu mũi tên: Dòng chảy của chương trình
* Ký hiệu mũi tên + hình tròn nối: Nối hai trang kế tiếp nhau

## Câu 8: Vẽ giải thuật giải phương trình bậc nhất?

Giải thuật phương trình bậc nhất dạng ax + b = 0 như sau:

BEGIN

INPUT a, b

IF a = 0 THEN

IF b = 0 THEN

DISPLAY “Phương trình vô số nghiệm”

ELSE

DISPLAY “Phương trình vô nghiệm”

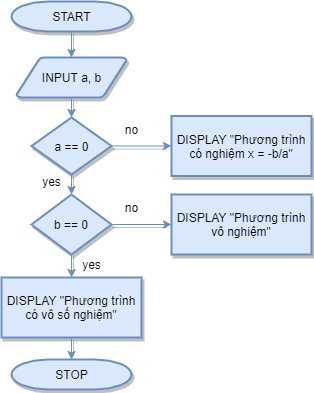
END IF

ELSE

DISPLAY “Phương trình có nghiệm x = -b/a

END IF

END



## Câu 9: Bạn biết gì về VCS – Version Control System (hệ thống quản lý phiên bản)?

Version Control System (VCS) – Hệ thống quản lý phiên bản mã nguồn là một phần mềm hỗ trợ:

* 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 xóa mất

## Câu 10: Phân biệt hệ thống quản lý mã nguồn tập trung và hệ thống quản lý mã nguồn phân tán?

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 gì? Tại sao cần sử dụng git?

Git là một hệ thống điều khiển phiên bản (version control system) theo hình thức phân tán.

Git được sử dụng để quản lý mã nguồn (source code) và ghi nhận các thay đổi

**Git sẽ giúp bạn:**

* Lưu trữ mã nguồn tập trung
* Chia sẻ mã nguồn giữa các bên
* Cộng tác giữa các thành viên trong nhóm phát triển
* Khôi phục mã nguồn về các phiên bản khác nhau
* Dễ dàng chỉnh sửa mã nguồn
* Tránh trùng lặp, xung đột mã nguồn

## Câu 12: Git thuộc hệ thống quản lý mã nguồn tập trung hay phân tán?

Git thuộc hệ thống quản lý mã nguồn phân tán.

## Câu 13: Repository là gì? Phân biệt LocalRepository và RemoteRepository?

Repository thường được gọi ngắn gọn là Repo là nơi chứa toàn bộ các file mã nguồn và lịch sử của các file đó.

Có 2 loại Repository là Local Repository và Remote Repository.

**Local Repository:** Ở trên máy của lập trình viên.

**Remote Repository**: Ở trên một máy chủ chia sẻ (chẳng hạn như GitHub).

## Câu 14: Đồng bộ giữa các repository (local và remote) để làm gì?

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: Thao tác push của git để làm gì? Cú pháp câu lệnh?

git push: Đẩy mã nguồn từ Local Repository lên Remote Repository.

Cú pháp: git push -u origin master

## Câu 16: Thao tác pull của git để làm gì?

git pull: Cập nhật mã nguồn từ một Remote Repository về Local Repository.

## Câu 17: Thao tác clone của git để làm gì? Cú pháp câu lệnh?

git clone: Sao chép một Remote Repository về máy của lập trình viên.

Cú pháp: git clone https://github.com/USERNAME/REPOSITORY.git

## Câu 18: Nêu tên và ý nghĩa các câu lệnh cơ bản đã được học để làm việc với git?

git clone Sao chép một Repository

git init Khởi tạo một Repository.

git remote add origin https://github.com/user/repo.git Kết nối tới Remote Repository

git add <file\_name> Đưa một file vào quản lý trong chỉ mục của GIT (staged)

git commit -m "First commit" Commit phiên bản kèm theo Message

git push -u origin master Đồng bộ từ Local Repository tới Remote Repository

## Câu 19: www là gì?

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 và truy cập thông qua Internet.

## Câu 20: HTML là gì?

HTML (Hypertext Markup Language) là mã được dùng để xây dựng nên cấu trúc và nội dung của trang web. HTML không phải là một ngôn ngữ lập trình; nó là một ngôn ngữ đánh dấu (markup language) định hình cấu trúc nội dung.

## Câu 21: Nêu tên các thẻ làm việc với văn bản trong tài liệu html?

Tiêu đề (thẻ h1–h6): Thường chỉ sử dụng tối đa là <h1>–<h4>:

<h1>My main title</h1>

<h2>My top level heading</h2>

<h3>My subheading</h3>

<h4>My sub-subheading</h4>

Đoạn văn (thẻ p): <p>This is a single paragraph</p>

Danh sách (thẻ ol, ul): Các loại danh sách phổ biến nhất có thứ tự <ol> và danh sách không có thứ tự <ul>.

Liên kết (thẻ a): <a> — "a" viết tắt của "anchor". Để làm cho văn bản trong đoạn văn của bạn thành một liên kết

## Câu 22: Thẻ div dùng để làm gì? <div> khác gì <span>

Thẻ <div> hay thẻ div trong HTML được dùng để nhóm nhiều phần tử HTML lại với nhau thành một khối.

<div> có kiểu hiển thị là Block (khối), trong khi <span> là một thẻ có kiểu hiển thị là Inline (một dòng) để nhóm các văn bản.

## Câu 23: URL là gì? Phân biệt url tuyệt đối và url tương đối?

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 gồm 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: https://www.google.com.vn/search?q=URL

URL tương đối: search?q=URL

## Câu 24: Siêu liên kết là gì?

Hyperlink có nghĩa là một siêu liên kết, 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: Muốn liên kết đến một phần trên cùng trang thì làm thế nào?

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

Ví dụ:

<div id="top"></div>

<a href="#top">Go to top</a>

## Câu 26: Muốn liên kết đến một phần trên trang khác thì làm thế nào?

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

Ví dụ: <a href="https://www.google.com.vn/">Go to Google</a>

## Câu 27: Phân biệt internal link và external link?

Internal Link là một liên kết nội bộ từ trang này sang trang khác trên cùng một tên miền.

External link là liên kết ra bên ngoài trang web khác.

## Câu 28: Có mấy loại danh sách?

Trong HTML có 3 kiểu danh sách (HTML Lists) đó là có danh sách thứ tự (ordered list), danh sách không có thứ tự (Unordered list) và danh sách mô tả (description list). Cụ thể:

1. **An ordered HTML list**: Một danh sách có thứ tự có thể là số hoặc theo thứ tự bảng chữ cái.

<ol>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

1. **An unordered HTML list**: Danh sách không có thứ tự dành cho các danh sách có thứ tự của các vật phẩm, chẳng hạn như công thức

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

1. **Description Lists**:

<dl>

<dt>Coffee</dt>

<dd>- black hot drink</dd>

<dt>Milk</dt>

<dd>- white cold drink</dd>

</dl>

## Câu 29: Muốn hiển thị danh sách có thứ tự bắt đầu C. D. E… thì làm thế nào?

Dùng thẻ <ol> với thuộc tính type="A" và start="3"

<ol start="3" type="A">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

## Câu 30: Form hay biểu mẫu dùng để làm gì?

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

## Câu 31: Thuộc tính action của form dùng để làm gì?

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: Phân biệt 2 method là get và post?

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: Phân biệt các nút bấm có type là: button, submit, reset?

<input type="button" value="Anonymous button"> <!-- Không có hiệu ứng khi nhấn nhưng có thể được tùy chỉnh bằng JavaScript -->

<input type="submit" value="Submit"> <!-- Gửi dữ liệu form đến máy chủ. -->

<input type="reset" value="Reset"> <!-- Đặt lại tất cả giá trị form về mặc định. -->

## Câu 34: Muốn submit form đến trang CodeGym.vn thì làm thế nào?

<form action="https://codegym.vn" method="get">

<input type="submit" value="Submit">

</form>

## Câu 35: Nếu muốn nối 3 ô trên cùng 1 hàng thì làm thế nào?

Sử dụng thuộc tính colspan

Ví dụ: <td colspan="3"></td>

## Câu 36: Nếu muốn nối 2 ô trên cùng 1 cột thì làm thế nào?

Sử dụng thuộc tính rowspan

Ví dụ: <td rowspan="3"></td>

## Câu 37: Phân biệt thuộc tính cellpadding và thuộc tính cellspacing?

* 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: Trình bày các cách nhúng JavaScript vào 1 tài liệu html?

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: Phát biểu “JavaScript phân biệt chữ hoa chữ thường ” đúng hay sai?

JavaScript phân biệt hoa thường — myVariable và myvariable là hai biến khác nhau

## Câu 41: Trình bày về quy tắc CamelCase và camelCase?

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 gì? Phân biệt biến cục bộ (local) và biến toàn cục (global)?

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. (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: Phân biệt: var, let, const?

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: Phát biểu “JavaScript là ngôn ngữ định kiểu yếu hay động” nghĩa là gì? Trình bày?

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.

let foo = 42; // foo có kiểu dữ liệu number

foo = 'bar'; // foo bây giờ có kiểu dữ liệu string

foo = true; // foo bây giờ có kiểu dữ liệu boolean

## Câu 45: Nêu tên các kiểu dữ liệu nguyên thủy trong JavaScript?

Trong JS có 6 kiểu dữ liệu nguyên thuỷ (Primitive Data):

* *Number*
* *Boolean*
* *String*
* *Null*
* *Undefined*
* *Symbol*

## Câu 46: Trình bày về toán tử typeof? Khi khai báo “let x=10;” thì x có kiểu là gì?

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

let x = 10; // x có kiểu dữ liệu là number

console.log(typeof(x)) // "number"

## Câu 47: Trình bày các cách đưa thông báo trong JavaScript?

Có 3 cách đưa thông báo trong JS:

1. alert(): Dùng để hiển thị một hộp thoại thông báo đến người dùng
2. 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)
3. prompt(): Dùng để hiển thị một hộp thoại thông báo và yêu cầu người dùng nhập dữ liệu vào.

## Câu 48: Trình bày về các loại toán tử trong JavaScript?

Toán tử cộng/nối chuỗi

Dùng để cộng hai số, hoặc nối hai chuỗi lại với nhau, ký hiệu +.

### Toán tử số học

Toán tử xuất hiện ở giữa hai toán hạng, cho phép thực hiện các phép tính với giá trị số và chuỗi và trả về một giá trị duy nhất. Các toán tử bao gồm: + (cộng),- (trừ), \* (nhân), / (chia), % (chia lấy dư).

### Toán tử tăng, và giảm

Chúng chỉ thực hiện được trên một toán hạng duy nhất. Toán tử tăng làm tăng giá trị lên 1, trong khi toán tử giảm làm giảm giá trị xuống 1; các toán tử có thể đặt trước hoặc sau toán hạng. Các toán tử bao gồm: ++ (tăng), -- (giảm).

### Toán tử quan hệ

Là các toán tử dùng để so sánh giữa hai toán hạng. Sau khi thực hiện một so sánh, chúng trả lại một giá trị true (đúng) hay false (sai). Các toán tử bao gồm: == (bằng nhau), != (khác nhau), === (bằng nhau và cùng kiểu dữ liệu), !== (khác nhau và khác kiểu dữ liệu), > (lớn hơn, < (nhỏ hơn), >= (lớn hơn hoặc bằng), <= (nhỏ hơn hoặc bằng).

### Toán tử luận lý (logic)

Thực hiện các phép toán logic trên hai toán hạng. Chúng thuộc loại toán tử quan hệ, vì chúng trả về một giá trị boolean. Các toán tử bao gồm: && (và), || (hoặc), ! (phủ định).

### Toán tử đặc biệt (Toán tử ba ngôi)

Là toán tử điều kiện còn được biết đến với toán tử 3 ngôi. Cú pháp của toán tử này như sau.

**Syntax (cú pháp)**:

variable\_name = (condition) ? exprIfTrue : exprIfFalse

Toán tử này sẽ trả lại giá trị là kết quả của exprIfTruenếu điều kiện có giá trị boolean bằng true, ngược lại nó sẽ trả lại giá trị bằng biểu exprIfFalse.

## Câu 49: Phân biệt giữa 2 toán tử == và ===?

==: So sánh hai giá trị với nhau.

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

## Câu 50: Phân biệt giữa 2 toán tử && và || trong JavaScript?

&& (và): Trả về giá trị truekhi tất cả các toán hạng đều truevà falsekhi ít nhất một toán hạng false.

|| (hoặc): Trả về giá trị truekhi ít nhất một toán hạng truevà falsekhi tất cả các toán hạng false.

## Câu 51: Kết quả biểu thức +0===-0 là true hay false?

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

## Câu 52: Trình bày về toán tử ++, -- và – trong JavaScript?

Toán tử ++ dùng để tăng giá trị thêm 1 đơn vị.

Toán tử -- dùng để giảm giá trị đi 1 đơn vị.

Toán tử - dùng để thao tác với số đối.

## Câu 53: Phân biệt ++a và a++?

a++ trả về giá trị của a trước khi tăng.

++a tăng giá trị của a trước rồi mới trả về giá trị sau khi tăng.

## Câu 54: Câu lệnh điều kiện/rẽ nhánh là gì?

Câu lệnh điều kiện là cấu trúc mã nguồn cho phép bạn kiểm tra xem liệu một biểu thức có trả về true hay không, rồi chạy các đoạn mã nguồn kế tiếp dựa theo kết quả ấy.

## Câu 55: Có những loại câu lệnh điều kiện nào?

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

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

## Câu 56: Trình bày về câu lệnh if?

Sử dụng if nếu điều kiện có giá trị đúng (true) thì khối mã bên trong if ({ ... }) sẽ được thực thi.

Nếu điều kiện có giá trị sai (false) thì khối mã bên trong if sẽ được bỏ qua. Luồng thực thi sẽ chuyển xuống ngay sau khối lệnh if.

**Syntax (cú pháp):**

if (condition) {

// block of code to be executed if the condition is true

}

Trong đó:

• condition: Là biểu thức trả về giá trị kiểu boolean.

• statements: Các câu lệnh sẽ được thực thi nếu điều kiện trả về true.

## Câu 57: Trình bày về câu lệnh switch?

switch-case là một cấu trúc điều kiện cho phép lựa chọn thực thi các khối lệnh khác nhau dựa trên kết quả của việc so sánh.

switch-case so sánh giá trị của một biến với lần lượt từng giá trị một, nếu có giá trị phù hợp với biến thì khối lệnh tương ứng sẽ được thực thi.

switch-case không thể thay thế if-else trong tất cả các trường hợp.

switch (expression) {

case choice1:

// run this code

break;

case choice2:

// run this code instead

break;

// include as many cases as you like

default:

// actually, just run this code

}

## Câu 58: Tại sao lại nói else là một tùy chọn (optional) của câu lệnh if?

Vì nếu thiếu else thì chương trình vẫn hoạt động bình thường mà không hề phát sinh lỗi.

## Câu 59: Khi nào 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: Câu lệnh break dùng để làm gì? Nếu một case mà không có break thì chương trình chạy thế nào?

Câu lệnh break sẽ thoát ra khỏi khối mã trong switch hoặc trong vòng lặp.

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 có bắt buộc trong câu lệnh switch?

Không bắt buộc.

## Câu 62: Trình bày về toán tử 3 ngôi?

Toán tử 3 ngôi (Conditional (Ternary) Operator) là toán tử duy nhất nhận 3 toán hạng. Toán tử này là cách viết tắt của câu lệnh điều kiện if-else.

**Syntax (cú pháp)**:

variable\_name = (condition) ? exprIfTrue : exprIfFalse

Toán tử này sẽ trả lại giá trị là kết quả của exprIfTruenếu điều kiện có giá trị boolean bằng true, ngược lại nó sẽ trả lại giá trị bằng biểu exprIfFalse.

## Câu 63: Trình bày về sự kiện và trình quản lý sự kiện trong JavaScript?

Sự kiện (Event) là sự tương tác của **JavaScript** với **HTML** được xử lý thông qua các sự kiện xảy ra khi người nhấn một nút trên bàn phím hoặc thao tác bấm chuột hoặc trình duyệt thao tác với trang web.

## Câu 64: Sự kiện onchange phát sinh khi nào?

Sự kiện onchange xảy ra khi giá trị (value) của thẻ **HTML** đổi. Thường dùng trong các thẻ form input hoặc select.

## Câu 65: Sự kiện onclick phát sinh khi nào?

Sự kiện onclick xảy ra khi click vào thẻ **HTML**.

## Câu 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?

Dùng hàm confirm() để hiển thị hộp thoại kèm message.

**Ví dụ:**

<button onclick="confirm('Bạn có chắc chắn muốn xóa không?'); return false;">Delete</button>

## Câu 67: Phân biệt onmousevover và onmouseout, onmousemove?

*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: Phân biệt onkeyup, onkeydown và onkeypress?

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

*onkeydown* : Xảy ra khi gõ một phím bất kỳ

*onkeypress* : Xảy ra khi nhấn giữ một phím

## Câu 69: Phân biệt onfocus và onblur?

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

*onblur* : Khi người dùng rời khỏi một phần tử trên *form*

## Câu 70: Sự kiện onload phát sinh khi nào?

*onload* : Xảy ra khi trang được mở trên trình duyệt và load đến phần tử body trong HTML

## Câu 71: Phân biệt innerHTML và innerText?

*innerHTML* : Chèn một đoạn mã HTML

*innerText* : Chèn một đoạn text đơn thuần

## Câu 72: Nếu muốn vô hiệu hóa (disabled) một button khi click vào 1 checkbox thì làm thế nào?

Đặt sự kiện *onclick* vào *checkbox*; Sau khi *checkbox* được click và thay đổi trạng thái *checked = true* thì thay đổi thuộc tính của *button disabled = true*

## Câu 73: Phân biệt thuộc tính readonly và disabled của html?

*readonly* giá trị trong thẻ *input,* chỉ hiển thị (không thể thay đổi) nhưng vẫn được sử dụng giá trị.

*disabled* vô hiệu hóa một thẻ *input hoặc button*, chỉ hiển thị (không thể thay đổi) nhưng không sử dụng hay nhấn vào được.

## Câu 74: Vòng lặp là gì?

Vòng lặp 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: Nêu các loại vòng lặp cơ bản trong JavaScript?

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* - lặp qua một khối lệnh trong khi biểu thức điều kiện đúng

## Câu 76: Phân biệt vòng lặp for,while,do…while?

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

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

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

## Câu 77: Khi nào nên sử dụng vòng lặp do…while?

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: Khi nào nên sử dụng vòng lặp for?

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

## Câu 79: Khi nào nên sử dụng vòng lặp while?

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

## Câu 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, dùng vòng lặp *do-while*.

## Câu 81: Yêu cầu người dùng tính tổng các số chẵn từ 10 đến 100 thì nên dùng vòng lặp loại nào?

Dùng vòng lặp *for*.

## Câu 82: Trình bày về vòng for thiếu?

Có thể bỏ qua cả 3 khối. Nhưng phải đảm bảo sử dụng câu lệnh *break* để kết thúc vòng lặp và một biến tăng giá trị để vòng lặp *for* chạy đến khi gặp câu lệnh *break*.

Ví dụ:

let i = 0;

for (;;) {

if (i > 3) break;

console.log(i);

i++;

}

## Câu 83: Nếu muốn hiển thị các cặp số nguyên dương có tổng là 10 thì viết thế nào?

for (let i = 1, j = 9; i <= j; i++, j--) {

console.log(i + *" "* + j);

}

## Câu 84: Nếu muốn hiển thị hình tam giác vuông \* thì nên viết thế nào?

for (let i = 0; i < 5; i++) {

for (let j = 0; j <= i; j++) {

document.write(*"\*"*);

}

document.write(*"<br>"*);

}

## Câu 85: Trình bày về cách chạy của vòng for lồng nhau?

Thường để duyệt mảng 2 chiều (two-dimensional array). Giả sử tạo bàn cờ Tic-tac-toe 3x3 thì vòng lặp *for* bên ngoài lặp 3 lần với biến *i < board.length*, vòng *for* lồng bên trong lặp 3 lần với biến *j < board[i].length*.

let board = [

['\*','\*','\*'],

['\*','\*','\*'],

['\*','\*','\*']

];

for (let i = 0; i < board.length; i++) {

for (let j = 0; j < board[i].length; j++) {

document.write(board[i][j]);

}

document.write(*"<br>"*);

}

## Câu 86: Phân biệt giữa câu lệnh break và continue?

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: 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);

}

Câu 88: Mảng là gì?

Mảng là một loại biến đặc biệt, có thể lưu được nhiều giá trị thay vì chỉ một giá trị như các biến thông thường

## Câu 89: Trình bày cú pháp khai báo mảng trong JavaScript?

Có hai cách 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 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.

## Câu 91: Kích thước của mảng là gì?

Số lượng các phần tử của mảng

## Câu 92: Phân biệt push() và unshift()?

push() : Thêm phần tử vào cuối mảng

unshift() : Thêm phần tử vào đầu mảng

## Câu 93: Phân biệt push() và pop()?

push() : Thêm phần tử vào cuối mảng

pop() : Xóa phần tử cuối mảng

Câu 94: Phân biệt shift() và unshift()?

shift() : Xóa phần tử đầu mảng

unshift() : Thêm phần tử vào đầu mảng

## Câu 95: Nếu muốn xóa phần tử cuối cùng của mảng dùng hàm nào?

Hàm pop()

## Câu 96: Nếu xóa phần tử đầu tiên của mảng dùng hàm nào?

Hàm shift()

## Câu 97: Muốn sắp xếp mảng thì dùng hàm nào?

Muốn sắp xếp mảng, ta dùng hàm sort()

## Câu 98: Có một mảng gồm tên của các bạn học viên. Nếu muốn sắp xếp tên này theo thứ tự alphabet làm thế nào?

let items = ['Tuấn Anh', 'Thành', 'Phương', 'Việt'];

items.sort(function(a, b) {

let nameA = a.toUpperCase(); // bỏ qua hoa thường

let nameB = b.toUpperCase(); // bỏ qua hoa thường

if (nameA < nameB) {

return -1;

}

if (nameA > nameB) {

return 1;

}

// name trùng nhau

return 0;

});

## Câu 99: Thao tác duyệt mảng là gì? Triển khai thao tác bằng code?

Duyệt qua tất cả các phần tử của mảng để lấy giá trị hoặc chỉ số

let numbers = [1, 2, 3];

for (let i = 0; i < numbers.length; i++) {

document.write(numbers[i]);

}

## Câu 100: Thao tác duyệt mảng 2 chiều?

Mảng hai chiều sử dụng 2 vòng lặp lồng nhau để duyệt mảng; cần chỉ số n hàng và m cột để xác định vị trí của phần tử mảng.

let numbers = [[1, 2], [3, 4], [5, 6]];

for (let i = 0; i < numbers.length; i++) {

for (let j = 0; j < numbers[i].length; j++) {

console.log(numbers[i][j]);

}

}

## Câu 101: Mảng 2 chiều cần mấy chỉ số để xác định duy nhất 1 phần tử?

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

## Câu 102: Viết code tính tổng các phần tử của ma trận số (ý tưởng)?

Tạo một biến sum để tính tổng các phần tử và gán giá trị ban đầu = 0. Sử dụng 2 vòng lặp for lồng nhau để duyệt các phần tử trong mảng. Trong vòng for thứ hai lấy biến sum += giá trị của phần tử.

Sau khi chạy xong hai vòng for thì in ra biến sum.

let numbers = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

let sum = 0;

for (let i = 0; i < numbers.length; i++) {

for (let j = 0; j < numbers[i].length; j++) {

sum += numbers[i][j];

}

}

console.log(sum);

## Câu 103: Viết code tính tổng các ô trên cùng hàng của ma trận số (ý tưởng)?

Dùng 2 vòng lặp for lồng nhau để duyệt mảng. Trong vòng for thứ nhất tạo một biến sum gán giá trị = 0 để tính tổng. Trong vòng for thứ hai lấy biến sum += giá trị của phần tử.

In ra biến sum trong cuối câu lệnh for thứ nhất.

let numbers = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

for (let i = 0; i < numbers.length; i++) {

let sum = 0;

for (let j = 0; j < numbers[i].length; j++) {

sum += numbers[i][j];

}

console.log(sum);

}

## Câu 104: Viết code tính tổng các ô trên cùng cột của ma trận số (ý tưởng)?

Dùng 2 vòng lặp for lồng nhau để duyệt mảng. Trong vòng for thứ nhất chạy đến array[0].length; tạo một biến sum gán giá trị = 0 để tính tổng.

Trong vòng for thứ hai chạy đến array.length; lấy biến sum += giá trị của phần tử array[j][i].

In ra biến sum trong cuối câu lệnh for thứ nhất.

let numbers = [[1, 2, 3], [4, 5, 6], [7, 8, 9]];

for (let i = 0; i < numbers[0].length; i++) {

let sum = 0;

for (let j = 0; j < numbers.length; j++) {

sum += numbers[j][i];

}

console.log(sum);

}

## Câu 105: Hàm (function) là gì?

Hàm (function) là một nhóm các câu lệnh thực hiện một nhiệm vụ nhất định. Nếu sử dụng hàm, mã nguồn sẽ trở nên ngắn gọn hơn và giúp tái sử dụng mã nguồn.

## Câu 106: Phân biệt giữa tham số hình thức (formal arguments) và tham số thực (actual arguments)?

Tham số hình thức (formal parameter) là các biến được định nghĩa trong hàm.

Tham số thực (actual parameter) hoặc đối số (argument) là các giá trị được truyền vào khi gọi hàm.

## Câu 107: Gọi hàm (invoke) là gì?

Invoke là cách để thực thi một hàm đã được định nghĩa trước đó.

## Câu 108: Phân biệt hàm gọi (calling function) và hàm được goi (called function)?

Calling Function (Gọi hàm) là sử dụng hàm đã được định nghĩa với các tham số nếu có.

Called Function (Định nghĩa một hàm) hay còn gọi là khai báo hàm - Là đặt tên cho hàm và chỉ định những việc cụ thể sẽ làm khi hàm đó được gọi.

## Câu 109: Phân biệt truyền tham trị và truyền tham chiếu?

|  |  |
| --- | --- |
| **Truyền tham trị** | **Truyền tham chiếu** |
| Một bản sao giá trị của biến được truyền vào hàm | Một địa chỉ ô nhớ của biến được truyền vào hàm |
| Những thay đổi trong hàm được giới hạn trong hàm, không làm thay đổi giá trị của biến được truyền vào hàm | Những thay đổi không chỉ giới hạn trong hàm mà còn làm thay đổi giá trị của biến được truyền vào hàm nếu trong hàm cũng làm thay đổi giá trị biến đó |
| Đối số trong hàm và tham số chính thức được tạo tại hai vị trí bộ nhớ khác nhau | Đối số trong hàm và tham số chính thức được tạo tại cùng một vị trí bộ nhớ |

## Câu 110: Câu lệnh return dùng để làm gì?

Dùng để trả về một giá trị của hàm.

## Câu 111: Có được phép viết “return;”?

Có.

## Câu 112: Phân biệt return và break?

break: Dừng một vòng lặp hoặc câu lệnh switch-case.

return: Dừng toàn bộ hàm và có thể trả về một giá trị của hàm.

## Câu 113: Hàm thực hiện việc kiểm tra nên trả về giá trị không? Nếu có, nên trả về gì?

Có; trả về true nếu việc kiểm tra điều kiện là đúng hoặc false.

## Câu 114: Tại sao hàm hoán vị swap(a, b) lại không thực hiện được việc hoán vị của các tham số gốc?

Giá trị được truyền vào hàm swap(a, b) là các tham số hình thức (formal parameter). Nếu hàm swap(a, b) thay đổi giá trị của tham số a và b. Nó sẽ không thay đổi giá trị của tham số gốc (Tham số thực - actual parameter) được truyền vào ở phạm vi bên ngoài hàm.

## Câu 115: Trình bày về OOP?

OOP (Object Oriented Programming) hay còn gọi là ngôn ngữ lập trình hướng đối tượng, là một phương pháp thiết kế và phát triển phần mềm dựa trên kiến trúc lớp (class) và đối tượng (object). Sau đây là mô tả về OOP:

* 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.
* 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 (attribute hoặc properties): Bao gồm các biến, các hằng, hay tham số của lớp đó để mô tả đặc tính của đối tượng
* Phương thức (method): Mô tả các hành vi của đối tượng (hoặc của lớp)

Các đối tượng có thể có quan hệ với nhau

## Câu 116: Phân biệt giữa lập trình hướng đối tượng và lập trình hướng thủ tục/chức năng?

**Ngôn ngữ Thủ tục**

* Hướng tiếp cận: từ trên xuống (top down)
* 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

**Ngôn ngữ Hướng đối tượng (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: Phân biệt giữa object và class?

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).

Class là khuôn mẫu chứa những khai báo định dạng và nguyên tắc hoạt động.

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).

## Câu 118: Hai khái niệm “Môn học” và “Môn học JavaScript” thì đâu là object?

“Môn học” là class; “Môn học JavaScript” là object thuộc class “Môn học”.

## Câu 119: Mô tả về sơ đồ lớp?

Sơ đồ lớp mô tả những đặc điểm khái quát nhất về lớp bao gồm: Tên lớp (class), danh sách các thuộc tính (properties) và các phương thức (method).

## Câu 120: Vẽ sơ đồ lớp của lớp Học viên?

|  |
| --- |
| Student |
| name: string  age: number  class: string  gender: boolean |
| readAnything(): void  writeAnything(): void  speakAnything(): void |

## Câu 121: Thuộc tính là gì? Nên dùng từ loại nào để biểu diễn thuộc tính?

Property (thuộc tính) là những đặc điểm đặc trưng của đối tượng, thể hiện thông qua những giá trị cụ thể.

Nên dùng danh từ để biểu diễn thuộc tính.

## Câu 122: Phương thức là gì? Nên dùng từ loại nào để biểu diễn phương thức?

Phương thức (Method)/Hành vi (Behavior)/Hành động (Action)/Khả năng (Capability): 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: Cú pháp định nghĩa 1 class trong JavaScript?

Syntax (cú pháp):

**Sử dụng Object Constructors**

Trong ECMAScript 5 (ES5) trở về trước sử dụng từ khóa function

function Person(name, age) {

this.name = name || '';

this.age = age || 0;

this.getName = function() {

return this.name;

};

this.setName = function(value) {

this.name = value;

}

}

Từ ECMAScript 6 (ES6) trở đi sử dụng từ khóa class để định nghĩa một lớp và luôn thêm phương thức (method) constructor().

class Person {

constructor(name, age) {

this.name = name || '';

this.age = age || 0;

}

getName() {

return this.name;

}

setName(value) {

this.name = value;

}

}

**Sử dụng Object Literals**

Literals là một cách tạo đối tượng và mảng ngắn nhất trong JavaScript. Để tạo ra một đối tượng rỗng. VD:

const Person = {

name: "John Doe",

age : 25,

setName(value) {

this.name = value;

},

getName() {

return this.name;

}

};

## Câu 124: Cú pháp tạo 1 đối tượng từ class?

let objectName = new objectName(parameters);

let myName = new Person("Nguyễn Bá Tuấn Anh", 25);

## Câu 125: Trình bày một số class có sẵn trong JavaScript mà em biết?

* Date
* Array
* String
* Number
* Boolean
* Object

## Câu 126: Cú pháp truy cập đến 1 thuộc tính của đối tượng?

objectName.propertyName; // myName.name

## Câu 127: Cú pháp truy cập đến 1 phương thức của đối tượng?

objectName.methodName(parameters); // myName.setName("Tuấn Anh")

## Câu 128: Từ khóa this dùng để làm gì?

Từ khóa this để chỉ chính đối tượng đó, thuộc sở hữu của đối tượng đó.

## Câu 129: Trình bày về DOM (Document Object Model – Mô hình đối tượng tài liệu)?

DOM là một giao diện lập trình ứng dụng (API). DOM được dùng để truy xuất các tài liệu dạng HTML, có dạng một cây cấu trúc dữ liệu.

Trong JavaScript để thao tác được với các thẻ HTML ta phải thông qua đối tượng documnent (DOM).

## Câu 130: Thẻ <canvas> dùng để làm gì?

Canvas API cung cấp phương tiện để vẽ các hình ảnh đồ họa trong trang web thông qua JavaScript và HTML.

<canvas id="myCanvas" width="150" height="150"></canvas>

## Câu 131: Vẽ hình chữ nhật trên canvas?

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

ctx.fillStyle = 'green';

ctx.fillRect(10, 10, 150, 100);

## Câu 132: Vẽ hình tròn trên canvas?

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

ctx.fillStyle = 'red';

ctx.arc(35, 35, 20, 0, Math.PI \* 2, true);

ctx.fill();

## Câu 133: Vẽ text trên canvas?

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

ctx.font = "18px Arial";

ctx.fillStyle = 'black';

ctx.fillText("Hello World", 0, 100);

## Câu 134: Nếu muốn di chuyển một hình tròn trên canvas cần làm gì?

<canvas id="myCanvas" width="15 0" height="150"></canvas>

<script>

const canvas = document.getElementById('myCanvas');

const ctx = canvas.getContext('2d');

const ballRadius = 10;

let x = canvas.width / 2, y = canvas.height - 30, dx = 2; dy = -2;

setInterval(draw, 10);

function drawBall() {

ctx.beginPath();

ctx.arc(x, y, ballRadius, 0, Math.PI \* 2);

ctx.fillStyle = "#0095DD";

ctx.fill();

ctx.closePath();

}

function draw() {

ctx.clearRect(0, 0, canvas.width, canvas.height);

drawBall();

if (x > canvas.width-ballRadius || x < ballRadius) {

dx = -dx;

}

if (y > canvas.height-ballRadius || y < ballRadius) {

dy = -dy;

}

x += dx;

y += dy;

}