**BỘ CÂU HỎI PHỎNG VẤN MODULE 1**

**BOOTCAMP PREPARATION 2.0**

|  |  |
| --- | --- |
| **STT** | **Câu hỏi** |
| 1 | **Hệ thống các kiến thức đã học ở Module1**   * Psedo code & flowchart * Git & html * Html form & table * Tổng quan Js * Biến , kiểu dữ liệu và toán tử * Cấu trúc điều kiện * Cấu trúc lặp * Mảng * Hàm function * Class & object |
| 2 | **Giải thuật/Thuật toán là gì?**  Giải thuật/thuật toán là các bước chỉ thị để giải quyết một vấn đề |
| 3 | **So sánh việc trình bày giải thuật bằng pseudo-code và flowchart?**  Pseudo code:   * Sử dụng ngôn ngữ tự nhiên để biểu diễn thuật toán đó * Sử dụng ngôn ngữ tiếng anh như Begin, ENd, Start, Input-Output,...   Flowchart:   * Mô tả bằng sơ đồ khối , mỗi một khối quy định về một hành động cụ thể.   Vidu:   * hình elip: bắt đầu và kết thúc một ctrình * hình vuông mô tả một phép toán * hình thoi so sánh điều kiện rẻ nhánh * hình bình hành thực hiện các bước nhập và xuất * và các mũi tên chỉ dòng chảy |
| 4 | **Tại sao lại cần trình bày giải thuật trước khi bắt tay vào code?**   * Giúp ta hiểu rõ vấn đề bài toán để giải quyết cách dễ dàng * logic rõ ràng tránh bị sai sót * tối ưu hóa bài toán |
| 5 | **Tại sao lại nên sử dụng flowchart hơn pseudo-code khi thể hiện giải thuật của bài toán phức tạp?**   * Dễ đọc dễ hiểu vấn đề bài toán đã cho. * Hỗ trợ gỡ rắc rối trong việc thực hiện thuật toán. * Phân loại từng chức năng của nó thông qua sơ đồ khối |
| 6 | **Tại sao giải thuật lại cần hữu hạn các bước thực thi?**   * Tính khả thi của tính toán * Quản lí tài nguyên máy * Dự đoán và kiểm soát * Phân tích hiệu suất |
| 7 | **Trình bày ý nghĩa các ký hiệu sử dụng khi vẽ flowchart?**   * hình elip: bắt đầu và kết thúc một ctrình * hình vuông mô tả một phép toán * hình thoi so sánh điều kiện rẻ nhánh * hình bình hành thực hiện các bước nhập và xuất * và các mũi tên chỉ dòng chảy |
| 8 | **Vẽ giải thuật giải phương trình bậc nhất?**  **https://lh5.googleusercontent.com/eoG5iTdHzuwFSesgSkq38X4rET3wEQ1oMu8EJU9S2J2EGudKnaGMvH83KiB43jw50162PX6GJWPIScaW-XXoSF-a0N2C-TpglrO-180zbaAxXk-0kPaossSM1qopGaI2Sp3zbTrfHUNGzXfbrix-X6k** |
| 9 | **Bạn biết gì về VCS – Version Control System (hệ thống quản lý phiên bản)?**  Hệ thống quản lí phiên bản : là một hệ thống được sử dụng để quản lí và theo dõi sự quản lí của mã nguồn và tài liệu trong quá trình phát triển phần mềm |
| 10 | **Phân biệt hệ thống quản lý mã nguồn tập trung và hệ thống quản lý mã nguồn phân tán?**  Hệ thống quản lí mã nguồn tập trung:   * + Kiến trúc máy tính   + Phân nhánh hạn chế   + Khó làm việc ngoại tuyến   Hệ thống quản lí mã nguồn phân tán:   * + Kiến trúc phân tán   + Phân tán mạnh mẽ   + Dễ làm việc ngoại tuyến |
| 11 | **Git là gì? Tại sao cần sử dụng git?**  - Git là một hệ thống quản lí mã nguồn và phát triển dự án phần mềm  - Vì:  + Quản lí phiên bản hiệu quả  + Phân nhánh và hợp nhất dễ dàng  + Làm việc đồng thời  + Làm việc ngoại tuyến  + Bảo mất và tính toàn vẹn |
| 12 | **Git thuộc hệ thống quản lý mã nguồn tập trung hay phân tán?**  - Git thuộc hệ thống quản lí mã nguồn phân tán |
| 13 | **Repository là gì? Phân biệt LocalRepository và RemoteRepository?**  - Repository ( thường được gọi là repo): là một nơi chứa toàn bộ máy nguồn  - Phân biệt Local repo và remote repo:  + Local Repository : ở trên máy người lập trình viên  + Remote Repository: Ở trên máy chủ chia sẽ ( vdu: GitHub) |
| 14 | **Đồng bộ giữa các repository (local và remote) để làm gì?**   * + Tối ưu hóa cộng tác   + Quản lí phiên bản   + Đảm bảo tính ổn định của dự án |
| 15 | **Thao tác push của git để làm gì? Cú pháp câu lệnh?**  - Thao tác push của git để đưa các thay đổi từ Local lên Remote  - Cú pháp câu lệnh: git push |
| 16 | **Thao tác pull của git để làm gì?**  - Thao tác pull dùng để cập nhập những thay đổi mới nhất từ Romote Repository về Local Repository |
| 17 | **Thao tác clone của git để làm gì? Cú pháp câu lệnh?**  - git clone: Sao chép một Repository |
| 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 add: Đưa các file vào trong vùng staged  • git commit: Ghi nhận các thay đổi  • git push: Đưa các thay đổi từ local repository lên remote repository |
| 19 | **www là gì?**  - www là (world wide web): Mạng toàn cầu |
| 20 | * **HTML là gì?**   - html(Hyper Text Makeup Language) : Ngôn ngữ đánh dấu siêu văn bản |
| 21 | **Nêu tên các thẻ làm việc với văn bản trong tài liệu html?**   * + <h>   + <p>   + <span>   + <br>   + <hr>   + <a> |
| 22 | **Thẻ div dùng để làm gì? <div> khác gì <span>**  - <div>: được sử dụng để tạo khu vực chứa , phân chia và cấu trúc trang wed  So sánh:   * + <div> : được sử dụng để tạo khu vực chứa, phân chia và cấu trúc trang web.   + <span> : sử dụng để tạo kiểu hoặc định dạng một phần nhỏ của văn bản |
| 23 | **URL là gì? Phân biệt url tuyệt đối và url tương đối?**  - URL là giao diện hoàn chỉnh một đường dẫn  - Phân biệt:   * + URL tuyệt đối cung cấp mọi chi tiết cần thiết để xác định vị trí tài nguyên một cách chính xác. * + URL tương đối chỉ định vị trí tài nguyên dựa trên vị trí liên quan đến trang hoặc thư mục hiện tại. |
| 24 | **Siêu liên kết là gì?**   * + Là một phần tử trong html và các hệ thống thông tin khác cho phép người dùng truy cập 1 trang bằng cách nhấp chuột vào |
| 25 | **Muốn liên kết đến một phần trên cùng trang thì làm thế nào?**  Các cách:   * + Sử dụng thẻ <a> và thuộc tính id   + Sử dụng Url cùng tên mảng |
| 26 | **Muốn liên kết đến một phần trên trang khác thì làm thế nào?**  Các cách:   * + Sử dụng url tuyệt đối   + Sử dụng url tương đối |
| 27 | **Phân biệt internal link và external link?**   * Internal link liên kết giữa các phần khác nhau của cùng một trang web hoặc trang trong cùng một tên miền. * External link liên kết đến các trang web hoặc tài nguyên nằm ngoài tên miền hiện tại. |
|  |  |

**28. Có mấy loại danh sách?**

* Kiểu không sắp xếp (*Unordered List=> ul*): 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. (Đánh theo thứ thự: Alphabet, số la mã, số tự nhiên 1,2,...)
* Kiểu sắp xếp *(Ordered List=>ol*): Là kiểu hiển thị một danh sách mà các mục con của nó được sắp xếp theo thứ tự bằng số hoặc chữ cái. ( đánh dấu: vòng tròn, dấu chấm, dấu hình vuông,..)

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

* Vd: <ul>

           <li> C </li>

           <li> D </li>

           <li> E </li>

     </ul>

**30. Form hay biểu mẫu dùng để làm gì?**

* Form (<form></form>):

+  Form không chứa dữ liệu đầu vào trực tiếp. Thay vào đó, nó chỉ định cách dữ liệu được gửi đi và xử lý bởi máy chủ.

+ Dùng để xác định phương thức gửi dữ liệu, vd như get, post -> có thuộc tính là Method , hoặc là dùng để chỉ định URL xử lý dữ liệu-> Action.

+ Vd đoạn code:

<form action=”/process\_data” method=”POST”>

// các TH nhập liệu sẽ được định nghĩa ở đây//

<input type=”text” name=”username”>

<input type=”passwork” name=”passwork”>

<input type=”submit” value=” submit”>

* Biểu mẫu:

+ Biểu mẫu là khái niệm rộng hơn, nó bao gồm cả form(phần tử HTML)

Và các trường dữ liệu bên trong form.

+ Biểu mẫu bao gồm các trường nhập liệu (inputfields) như ô văn bản, ô chọn, nút radio, nút checkbox,..để người dùng nhập vào và chọn dữ liệu.

(có thuộc tính là name(tên trường dữ liệu), type (loại trường nhập liệu), placeholder (gợi ý vb khi trống,...)).

+ Vd đoạn code như:

<input type=”text” name=”username” placeholder=”Enter your name”>

<input type=”password” name=” password” placeholder=”Enter your password”>

<input type=”submit” value=”submit”>

**31.** **Thuộc tính action của form dùng để làm gì?**

+ Dùng để chỉ định URL xử lý dữ liệu.

**32.** **Phân biệt 2 method là get và post?**

+ GET: là phương thức "truy vấn" và thường được sử dụng khi muốn hiển thị thông tin hoặc truy xuất dữ liệu từ máy chủ.

Dữ liệu được gửi qua URL, dễ nhìn thấy trong địa chỉ trình duyệt, có giới hạn về kích thước dữ liệu và không nên sử dụng khi gửi dữ liệu bảo mật.

+POST: Được sử dụng để gửi dữ liệu từ trình duyệt đến máy chủ thông qua phần thân của yêu cầu HTTP.

Dữ liệu được gửi ẩn danh và không xuất hiện trong URL, không giới hạn về kích thước dữ liệu và thích hợp sử dụng được khi gửi dữ liệu bảo mật.

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

+ Type button: thường không có hành động mặc định khi được nhấn. Tức là

không có hành động gửi dữ liệu hoặc đặt lại biểu mẫu xảy ra, trừ khi bạn xử lý nó bằng JavaScript.

+ Type submit: được sử dụng trong biểu mẫu để gửi dữ liệu của biểu mẫu đến máy chủ. Tức là người dùng nhấn vào nút này, biểu mẫu sẽ được gửi đến đích được xác định bởi thuộc tính action trong thẻ <form>.

+ Type reset: được sử dụng trong biểu mẫu để đặt lại giá trị của tất cả các trường nhập liệu trong biểu mẫu về giá trị ban đầu.

**34. Muốn submit form đến trang CodeGym.vn thì làm thế nào?**

+ Để submit form đến trang CodeGym.vn, bạn cần biết URL hoặc endpoint xử lý dữ liệu form, thường được xác định trong thuộc tính action của thẻ <form>

**35. Nếu muốn nối 3 ô trên cùng 1 hàng thì làm thế nào?**

+ Sử dụng phần tử block (div) hoặc phần tử inline (span) để nối các ô lại với nhau.

Vd:

Phần tử Block (div):

<div style="display: flex;">

<div>Ô 1</div>

<div>Ô 2</div>

<div>Ô 3</div> // dùng nếu đề yêu cầu nối 3 ô

</div>

(chúng ta đã sử dụng phần tử <div> với thuộc tính display: flex ,các ô con sẽ hiển thị trên cùng một hàng )

Phần tử inline(span):

<span>Ô 1</span>

<span>Ô 2</span>

<span>Ô 3</span> // dùng nếu đề yêu cầu nối 3 ô

(Các phần tử inline mặc định sẽ hiển thị trên cùng một hàng nếu không có thuộc tính hoặc CSS nào khác thay đổi)

**36. Nếu muốn nối 2 ô trên cùng 1 cột thì làm thế nào?**

+ Sử dụng phần tử block (div) hoặc phần tử inline (span) để nối các ô lại với nhau.

+ Nếu tạo 2 ô thì phần tử block <div> không cần thiết lập thuộc tính “display” vì mặc định, các phần tử block sẽ hiển thị dưới dạng các khối và sẽ nối hai ô lại với nhau theo hướng cột.

+ Phần tử <span> là phần tử inline. Để hiển thị các phần tử inline trên cùng một cột, chúng ta sử dụng thẻ <br> sau mỗi phần tử <span>.

**37. Phân biệt thuộc tính cellpadding và thuộc tính cellspacing?**

+ **Cellpadding:**  Thuộc tính Cellpadding được sd để điều chỉnh khoảng cách giữa nội dung của ô và viền của ô, giá trị là một số không âm, đo lường bằng đơn vị pixel (px).

+ **Cellspacing:**  Thuộc tính Cellspacing điều chỉnh khoảng cách giữa các ô trong bảng, giá trị là một số không âm, đo lường bằng đơn vị pixel (px).

**38. Trình bày các cách nhúng javascript vào 1 tài liệu html?**

+ Sử dụng thẻ <script> để viết mã JavaScript trực tiếp trong tài liệu HTML. (Nhúng trực tiếp)

+ Đặt mã JavaScript trong một tập tin riêng và nhúng nó vào tài liệu HTML bằng thẻ <script> với thuộc tính src để chỉ định đường dẫn tới tập tin JavaScript. (Nhúng từ một tập tin javacript bên ngoài)

+ Sử dụng các sự kiện HTML để gọi mã JavaScript, ví dụ: onclick, onload,..

**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. Vì JavaScript là một ngôn ngữ script có hỗ trợ đối tượng, được sử dụng để phát triển các ứng dụng Internet trên cả client-side (trình duyệt) và server-side (máy chủ).

**40. Phát biểu “JavaScript phân biệt chữ hoa chữ thường ” đúng hay sai?**

+Cóa, các biến, tên hàm, tên thuộc tính, và các ký tự trong mã JavaScript sẽ được xử lý khác nhau nếu có sự khác biệt về chữ hoa và chữ thường

**41. Trình bày về quy tắc CamelCase và camelCase?**

+ Đều là quy tắc đặt tên trong đó các từ được viết liền nhau và mỗi từ bắt đầu bằng một chữ cái in hoa (ngoại trừ từ đầu tiên)

+ CamelCase trong tên không có dấu cách hoặc ký tự đặc biệt, thường được sử dụng khi đặt tên lớp (class).

ví dụ: ThisIsCamelCase.

+ camelCase bắt đầu từ chữ cái thường ở từ đầu tiên, được sử dụng khi đặt tên biến, hàm trong các ngôn ngữ lập trình như JavaScrip

ví dụ: thisIsCamelCase.

**42. Biến là gì? Phân biệt biến cục bộ (local) và biến toàn cục (global)?**

+ Biến là một khái niệm trong lập trình dùng để lưu trữ giá trị. Được sử dụng để tham chiếu và lưu trữ các giá trị như số, chuỗi, đối tượng, hoặc mảng trong mã lệnh.

+ Biến cục bộ (local variable): được khai báo và sử dụng trong phạm vi của một khối mã cụ thể, chẳng hạn như bên trong một hàm.

+ Biến toàn cục nghĩa là nó có thể được truy cập và sử dụng từ bất kỳ phạm vi nào trong mã lệnh, thường được khai báo bên ngoài các hàm.

**43. Phân biệt: var, let, const?**

+ Let: Không được khai báo trùng tên.

Phạm vi sd của biến trong block scope và block scope được xd bởi dấu (;)

Phải khai báo trước khi sd.

+ Var: Có thể khai báo trùng tên.

Phạm vi sd toàn cục

Cho sd trc khi khai báo.

+ Const: Khai báo hằng số và gtri hằng số khôg thể thay đổi.

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

**+** Phát biểu "JavaScript là ngôn ngữ định kiểu yếu hay động" có nghĩa là JavaScript là một ngôn ngữ lập trình mà kiểu dữ liệu của một biến có thể thay đổi trong quá trình thực thi của chương trình, và các kiểu dữ liệu này có thể tự động chuyển đổi khi được sử dụng trong các biểu thức. Trong JavaScript, biến có thể được khai báo mà không cần chỉ định kiểu dữ liệu của nó. Kiểu dữ liệu của biến sẽ được xác định dựa trên giá trị đầu tiên được gán cho biến. Sau đó, kiểu dữ liệu có thể thay đổi nếu giá trị của biến được thay đổi.

**45. Nêu tên các kiểu dữ liệu nguyên thủy trong JavaScript?**

+ Có 6 loại: + Number: số nguyên và số thập phân

+ String: ký tự hoặc văn bản

+ Boolean: hai giá trị

+ Null: giá trị rỗng hoặc không tồn tại (nó không chứa bất kỳ giá trị nào)

+ Undefined: một biến đã được khai báo nhưng chưa được gán giá trị.

+ Symbol: sử dụng để định danh các thuộc tính của đối tượng

( vd: let sym = Symbol('mySymbol');)

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

+ Toán tử typeof được sử dụng để xác định kiểu dữ liệu của một biến hoặc một giá trị. Nó sử dụng typeof trước một biểu thức hoặc biến, nó sẽ trả về một chuỗi biểu thị kiểu dữ liệu của biểu thức hoặc giá trị đó.

**Vd:** let x = 10;

console.log(typeof x); // In ra "number"

let y = "Hello";

console.log(typeof y); // In ra "string"

let z = true;

console.log(typeof z); // In ra "boolean"

let w;

console.log(typeof w); // In ra "undefined"

**47. Trình bày các cách đưa thông báo trong JavaScript?**

+ Sử dụng hàm alert():

Hàm alert() được sử dụng để hiển thị một thông báo đơn giản với một nút "OK".

+ Sử dụng hàm confirm():

Hàm confirm() hiển thị một hộp thoại xác nhận với hai lựa chọn "OK" và "Cancel".

+ Sử dụng hàm prompt():

Hàm prompt() hiển thị một hộp thoại yêu cầu người dùng nhập một giá trị.

Người dùng có thể nhập giá trị và nhấn "OK" hoặc "Cancel".

**48. Trình bày về các loại toán tử trong JavaScript?**

**+ Toán tử số học:**

+: Phép cộng

-: Phép trừ

\*: Phép nhân

/: Phép chia

%: Phép chia lấy dư

**+ Toán tử gán:**

=: Gán giá trị từ phía bên phải sang biến bên trái.

+=: Cộng và gán.

-=: Trừ và gán.

\*=: Nhân và gán.

/=: Chia và gán.

%=: Chia lấy dư và gán.

**+ Toán tử so sánh:**

==: So sánh bằng giá trị.

===: So sánh bằng giá trị và kiểu dữ liệu.

!=: So sánh khác giá trị.

!==: So sánh khác giá trị và kiểu dữ liệu.

>: So sánh lớn hơn.

<: So sánh nhỏ hơn.

>=: So sánh lớn hơn hoặc bằng.

<=: So sánh nhỏ hơn hoặc bằng.

**+ Toán tử logic:**

&&: Toán tử AND logic.

||: Toán tử OR logic.

!: Toán tử NOT logic.

**+ Toán tử chuỗi (String):**

+: Toán tử nối chuỗi (concatenation). Dùng để nối các chuỗi lại với nhau.

**+ Toán tử ba ngôi (Ternary):**

condition ? expr1 : expr2: Đánh giá biểu thức condition. Nếu condition đúng, trả về expr1, ngược lại trả về expr2.

**+ Toán tử kiểu dữ liệu:**

typeof: Xác định kiểu dữ liệu của biểu thức hoặc biến.

**+ Toán tử định danh đối tượng:**

instanceof: Kiểm tra xem một đối tượng có là một thể hiện của một kiểu đối tượng hay không.

**49. Phân biệt giữa 2 toán tử == và ===?**

+ Toán tử == (So sánh bằng giá trị):

Toán tử == so sánh hai giá trị mà không xem xét kiểu dữ liệu của chúng.

Nếu hai giá trị có giống nhau về giá trị (mặc dù kiểu dữ liệu khác nhau), toán tử == sẽ trả về giá trị true.

+ Toán tử === (So sánh bằng giá trị và kiểu dữ liệu):

Toán tử === so sánh hai giá trị và kiểu dữ liệu của chúng.

Chỉ khi cả hai giá trị và kiểu dữ liệu đều giống nhau, toán tử === mới trả về giá trị true.

|  |  |
| --- | --- |
| 50 | **Phân biệt giữa 2 toán tử && và || trong JavaScript?**  - Toán tử ‘&&’ : được sử dụng để thự hiện AND logic giữa hai thức  - Toán tử ‘||’ : được sử dụng để thự hiện Or logic giữa hai thức |
| 51 | **Kết quả biểu thức +0===-0 là true hay false?**  - True |
| 52 | **Trình bày về toán tử ++,-- và – trong JavaScript?**  - Toán tử ++ : tăng lên 1 đơn vị  - Toán tử --: GIảm đi 1 đơn vị  - Toán tử - : Phủ định số |
| 53 | **Phân biệt ++a và a++?**   * + ++a: Tiền tố tăng lên 1 đơn vị   + a++ : Hậu tố tăng lên 1 đơn vị |

**Câu 55, 56, 57: Có 3 câu lệnh điều kiện**:

* If xác định 1 khối mã đc thực thi => đúng
* Else xác định 1 khối mã đc thực thi => sai
* Switch-case xác định khối mã luân phiên đc thực thi

**58. Tại sao lại nói else là một tùy chọn (optional) của câu lệnh if?**

- Trong câu lệnh if, else là tùy chọn vì nó chỉ được sử dụng khi một điều kiện không đúng.

- Nếu không có else, chương trình sẽ chỉ thực hiện câu lệnh trong khối if và bỏ qua các câu lệnh trong khối else.

**59. Khi nào nên sử dụng câu lệnh switch?**

- Câu lệnh switch nên được sử dụng khi có nhiều trường hợp xử lý khác nhau dựa trên giá trị của một biến hay một biểu thức.

- Switch thường được sử dụng khi số trường hợp cần xử lý là lớn và câu lệnh if-else không phù hợp.

**60. Câu lệnh break dùng để làm gì? Nếu một case mà không có break thì chương trình chạy thế nào?**

- Câu lệnh break được sử dụng để dừng vòng lặp hoặc câu lệnh switch.

- Nếu một case không có break, chương trình sẽ tiếp tục thực hiện các case tiếp theo trong câu lệnh switch cho đến khi gặp break hoặc kết thúc switch.

**61. Mệnh đề default có bắt buộc trong câu lệnh switch?**

- Mệnh đề default không bắt buộc trong câu lệnh switch, nó được sử dụng để xử lý các trường hợp không khớp với bất kỳ case nào trong switch.

**62. Toán tử 3 ngôi là gì?**

- Toán tử 3 ngôi là một toán tử có thể được sử dụng để viết một biểu thức rút gọn của câu lệnh if-else.

- Toán tử này có cú pháp: condition ? exprIfTrue : exprIfFalse. Nếu điều kiện condition đúng, biểu thức exprIfTrue được thực hiện, ngược lại biểu thức exprIfFalse được thực hiện.

**63. Sự kiện và trình quản lý sự kiện trong JavaScript là gì?**

- Sự kiện là một hành động được thực hiện bởi người dùng hoặc trình duyệt, ví dụ như nhấn nút chuột, nhập liệu vào form, hoặc tải trang.

- Trình quản lý sự kiện là một hàm JavaScript được gọi khi sự kiện xảy ra, và được sử dụng để xử lý các hành động liên quan đến sự kiện đó.

**64. Sự kiện onchange phát sinh khi nào?**

- Sự kiện onchange phát sinh khi giá trị của một phần tử HTML (ví dụ như input, select) thay đổi và mất trạng thái focus.

**65. Sự kiện onclick phát sinh khi nào?**

- Sự kiện onclick phát sinh khi người dùng nhấn chuột lên một phần tử HTML.

**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?**

- Để hiển thị hộp thoại xác nhận khi bấm vào nút delete, ta có thể sử dụng sự kiện onclick và hàm confirm() trong JavaScript.

**67. Phân biệt onmouseover và onmouseout, onmousemove?**

- Sự kiện onmouseover xảy ra khi con trỏ chuột di chuyển vào phần tử HTML, trong khi onmouseout xảy ra khi con trỏ chuột rời khỏi phần tử HTML. Sự kiện onmousemove xảy ra khi con trỏ chuột di chuyển trên phần tử HTML.

**68. Phân biệt onkeyup, onkeydown và onkeypress?**

- Sự kiện onkeydown xảy ra khi một phím được nhấn xuống, onkeyup xảy ra khi một phím được thả ra, và onkeypress xảy ra khi một phím được nhấn và giữ trong một khoảng thời gian.

**69. Phân biệt onfocus và onblur?**

- Sự kiện onfocus xảy ra khi một phần tử HTML nhận được trạng thái focus, tức là trở thành phần tử đang được tương tác, và onblur xảy ra khi phần tử đó mất trạng thái focus.

**70. Sự kiện onload phát sinh khi nào?**

- Sự kiện onload phát sinh khi trang web hoàn thành việc tải và hiển thị toàn bộ nội dung của trang.

**71. Phân biệt innerHTML và innerText?**

- Thuộc tính innerHTML là thuộc tính của một phần tử HTML và được sử dụng để lấy hoặc thiết lập nội dung HTML của phần tử đó. Trong khi đó, thuộc tính innerText được sử dụng để lấy hoặc thiết lập nội dung văn bản của phần tử đó, bỏ qua các thẻ HTML.

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

- Để vô hiệu hóa một button khi click vào một checkbox, ta có thể sử dụng sự kiện onclick và thuộc tính disabled trong JavaScript. Ví dụ:

**73. Phân biệt thuộc tính readonly và disabled của html?**

- Thuộc tính readonly được sử dụng để cho phép người dùng xem giá trị của một trường input nhưng không thể thay đổi giá trị đó. Trong khi đó, thuộc tính disabled được sử dụng để vô hiệu hóa một phần tử HTML, ngăn không cho người dùng tương tác với phần tử đó.

**74. Vòng lặp là gì?**

- Vòng lặp là một cấu trúc lập lại một khối mã trong một số lần cho đến khi một điều kiện được đáp ứng.

**75. Nêu các loại vòng lặp cơ bản trong JavaScript?**

- Các loại vòng lặp cơ bản trong JavaScript bao gồm vòng lặp for, while và do...while.

**76. Phân biệt vòng lặp for,while,do…while?**

- Vòng lặp for được sử dụng khi biết trước số lần lặp cần thực hiện. Vòng lặp while được sử dụng khi không biết trước số lần lặp cần thực hiện, và sẽ tiếp tục lặp lại cho đến khi điều kiện không còn đúng. Vòng lặp do...while cũng tương tự như vòng lặp while, tuy nhiên nó sẽ thực hiện ít nhất một lần trước khi kiểm tra điều kiện.

**77. Khi nào nên sử dụng vòng lặp do…while?**

- Vòng lặp do...while nên được sử dụng khi bạn muốn chắc chắn rằng khối mã trong vòng lặp sẽ được thực hiện ít nhất một lần, và sau đó sẽ tiếp tục lặp lại cho đến khi điều kiện

**78. Khi nào nên sử dụng vòng lặp for ?**

- Vòng lặp for thường được sử dụng khi chúng ta biết trước số lần lặp cần thiết để thực hiện một tác vụ. Vòng lặp này thường được sử dụng để lặp qua một mảng hoặc một chuỗi các phần tử. Vòng lặp for có thể được sử dụng để thực hiện các tác vụ như tính tổng các phần tử trong một mảng, tìm kiếm phần tử trong một mảng, hoặc sắp xếp các phần tử trong một mảng. Ví dụ:

let arr = [1, 2, 3, 4, 5];

let sum = 0;

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

sum += arr[i];

}

console.log(sum); // kết quả: 15

**79. Khi nào nên sử dụng vòng lặp while ?**

- Vòng lặp while thường được sử dụng khi chúng ta không biết trước số lần lặp cần thiết để thực hiện một tác vụ. Vòng lặp này thường được sử dụng để lặp lại cho đến khi một điều kiện cụ thể được đáp ứng. Vòng lặp while có thể được sử dụng để thực hiện các tác vụ như yêu cầu người dùng nhập dữ liệu, kiểm tra tính hợp lệ của dữ liệu nhập vào, hoặc xử lý lỗi trong chương trình.

let number = 0;

while (number <= 0) {

number = parseInt(prompt("Nhập vào một số lớn hơn 0: "));

}

console.log(number); // số nhập vào sẽ được hiển thị trên console

80. Có nên sử dụng vòng lặp để yêu cầu người dùng nhập đi nhập lại giá trị của một số cho đến khi số đó là số chẵn. Tùy thuộc vào cách hiện thực, nhưng trong trường hợp này, vòng lặp while hoặc do...while sẽ phù hợp hơn. Vì điều kiện lặp lại sẽ phụ thuộc vào giá trị của số nhập vào, và chúng ta cần lặp lại cho đến khi số đó là số chẵn. Ví dụ:

let number;

do {

number = parseInt(prompt("Nhập một số chẵn: "));

} while (number % 2 !== 0);

let number;

do {

number = parseInt(prompt("Nhập một số chẵn: "));

} while (number % 2 !== 0);

81. Để tính tổng các số chẵn từ 10 đến 100, ta có thể sử dụng vòng lặp for. Với vòng lặp này, ta có thể bắt đầu từ số 10 và lặp lại cho đến số 100, tăng giá trị của biến lặp lên 2 mỗi lần lặp để chỉ lặp qua các số chẵn.Ví dụ:

let sum = 0;

for (let i = 10; i <= 100; i += 2) {

sum += i;

}

console.log(sum); // kết quả: 2550

82. Vòng for thiếu, còn được gọi là vòng for-in, là một loại vòng lặp được sử dụng để lặp qua các thuộc tính của một đối tượng. Vòng lặp này lặp qua các thuộc tính có thể liệt kê được của một đối tượng, ví dụ như các thuộc tính của một đối tượng mảng hoặc các thuộc tính của một đối tượng định nghĩa bởi người dùng. Ví dụ:

for (variable in object) {

// code block to be executed

}

**83. Nếu muốn hiển thị cặp số nguyên dương từ 1-10 thì lm ntn ?**

115. OOP là viết tắt của Object-Oriented Programming, hay còn gọi là lập trình hướng đối tượng. Nó là một phương pháp lập trình trong đó các đối tượng được tạo ra để đại diện cho các thực thể trong thế giới thực. Các đối tượng này có các thuộc tính và phương thức, và các đối tượng có thể tương tác với nhau để thực hiện các nhiệm vụ cụ thể.

for (let i = 1; i < 10; i++) {

for (let j = 1; j < 10; j++) {

if (i + j === 10) {

console.log(i + " và " + j);

}

}

}

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

- Ý tưởng: Dùng 2 vòng lặp for lồng nhau, vòng lặp for ngoài dùng để duyệt các dòng từ trên xuống, vòng lặp for trong duyệt qua từng vị trí để vẽ dấu sao hoặc khoảng trắng tương ứng

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

- Vòng lặp "for" lồng nhau là cách sử dụng một hoặc nhiều vòng lặp "for" bên trong một vòng lặp "for" khác. Điều này cho phép bạn duyệt qua các phần tử của một mảng 2 chiều hoặc thực hiện các tác vụ lặp lặp theo các cấu trúc lồng nhau. Trong vòng lặp "for" lồng nhau, vòng lặp bên trong được thực hiện hoàn toàn trước khi vòng lặp bên ngoài tiếp tục.

|  |  |
| --- | --- |
| 86 | Phân biệt giữa câu lệnh break và continue?   * Câu lệnh break dùng để dừng vòng lặp ngay lập tức, và chuyển đến câu lệnh tiếp theo. Nếu break được dùng trong vòng lặp lồng, thì nó sẽ thoát khỏi vòng lặp đang chứa nó. // Trong switch-case, break dùng để kết thúc một case, nghĩa là nếu thực hiện xong case hợp lệ mà chưa gặp break thì nó sẽ thực hiện tiếp tục đến cuối cho đến khi gặp break      * Câu lệnh continue dùng để bỏ qua một bước lặp, nghĩa là nó sẽ bỏ qua tất cả những câu lệnh đằng sau nó và nhảy đến lần lặp tiếp theo lun     Khi câu lệnh "switch case" thực thi, nó sẽ kiểm tra các điều kiện case từ trên xuống dưới. Nếu tìm thấy một điều kiện case khớp với giá trị của biểu thức, các câu lệnh bên trong điều kiện case đó sẽ được thực hiện. Tuy nhiên, nếu không có điều kiện case nào khớp, chương trình sẽ thực hiện phần mã bên trong khối "default" (nếu có). Nếu không có khối "default", thì chương trình sẽ tiếp tục thực hiện các câu lệnh sau câu lệnh "switch". |
| 87 | Hiển thị các số không chia hết cho 5 từ 1-100 có sử dụng continue thì làm thế nào?  Ý tưởng: dùng vòng lặp for  for (var i = 0; i <= 100; i++) {      if ( i % 5 == 0 ) {         continue;      }      console.log(i)  } |
| 88 | Mảng là gì?   * Mảng là một dữ liệu thuộc kiểu tham chiếu ( reference type ), dùng để lưu trữ nhiều giá trị trong cùng một biến, mỗi giá trị sẽ là 1 phần tử của mảng * Khác vs biến tham trị, biến tham trị lưu trữ tên biến và giá trị trong cùng vùng nhớ Stack, thì biến mảng lưu trữ biến trong Stack, còn giá trị thì lưu trữ trong Heap |
| 89 | Trình bày cú pháp khai báo mảng trong JavaScript?  \_ Các cách thường dùng:   * var array = [];  ( khai báo mảng rỗng ) * var array = new Array(); (khai báo mảng rỗng )   Lưu ý var = new Array(5) ( khai báo mảng có 5 phần tử ) |
|  | 90 Chỉ số của mảng là gì? Chỉ số của mảng xuất phát từ 0 hay 1?  - Là các số nguyên không âm ( nghĩa là từ 0 trở lên ), dùng để đánh số chỉ mục và truy cập phần tử trong mảng. Quy ước chỉ số bắt đầu từ 0 ở phần tử đầu tiên, phần tử cuối cùng có chỉ số ( độ dài mảng – 1).  91 Kích thước của mảng là gì  - Chính là số phần tử của mảng. Để lấy kích thước mảng ta dùng thuộc tính length.  VD : var a = [1, 2, 3, -6, -12]  Console.log(a.length()); // 5  Trả về số nguyên ko âm, là độ dài mảng  92 Phân biệt push() vs unshift()  - Phương thức push cho phép thêm một hoặc nhiều phần tử vào cuối mảng  - push thay đổi kích thước của mảng và trả về chiều dài mới của mảng sau khi thêm  - Phương thức unshift cho phép thêm một hoặc nhiều phần tử vào đầu mảng, khi một phần tử được thêm thì các phần tử cũ sẽ được dịch ra đằng sau  - unshift cũng thay đổi kích thước mảng và trả về chiều dài mới của mảng sau khi thêm  93 Phân biệt push() vs pop()  - Phương thức push như đã nói ở trên  - Phương thức pop cho phép loại bỏ phần tử cuối cùng trong mảng và trả về chính phần tử được loại bỏ đó  - Nếu dùng pop trên một mảng rỗng nó sẽ trả về undefined  94 Phân biệt shift vs unshift  - Phương thức unshift như đã nói ở trên  - Phương thức shift cho phép loại bỏ phần tử đầu tiên trong mảng và trả về chính phần tử bị loại bỏ đó.  - shift cũng làm thay đổi kích thước của mảng  95 Muốn xóa phần tử cuối cùng của mảng ta dùng hàm nào?  - Dùng hàm pop  VD : var a = [1, 2, 3, 4, 5]  console.log(a.pop()); // 5  console.log(a); // [1, 2, 3, 4]  96 Muốn xóa phần tử đầu tiên của mảng ta dùng hàm nào?  \_ Dùng hàm shift  VD : var a = [1, 2, 3, 4, 5]  console.log(a.shift()); // 1  console.log(a); // [2, 3, 4, 5]  97 Muốn sắp xếp mảng dùng hàm nào?  - Dùng hàm sort(), hàm sort sẽ thay đổi mảng  98  99 Thao tác duyệt mảng là gì? Triển khai bằng code  - Duyệt bằng vòng lặp for  let arr = [1, 2, 3, 4, 5];  for (let i = 0; i < arr.length; i++) {  console.log(arr[i]); // In ra từng phần tử của mảng  }  - Duyệt bằng for…of  let arr = [1, 2, 3, 4, 5];  for (let element of arr) {  console.log(element); // In ra từng phần tử của mảng  }  - Duyệt bằng forEach()  let arr = [1, 2, 3, 4, 5];  arr.forEach(function(element) {  console.log(element); // In ra từng phần tử của mảng  });  - Duyệt bằng map()  let arr = [1, 2, 3, 4, 5];  let doubledArray = arr.map(function(element) {  return element \* 2; // Trả về mảng mới chứa giá trị gấp đôi của mỗi phần tử  });  console.log(doubledArray);  100 Thao tác duyệt mảng 2 chiều  - Dùng vòng lặp for lồng nhau, vòng lặp ngoài sẽ duyệt qua các phần tử của mảng cha ( mảng chứa các phần tử mảng con ), vòng lặp trong duyệt qua từng phần tử của mảng con  let twoDimArray = [  [1, 2, 3],  [4, 5, 6],  [7, 8, 9]  ];  for (let i = 0; i < twoDimArray.length; i++) {  for (let j = 0; j < twoDimArray[i].length; j++) {  console.log(twoDimArray[i][j]); // In từng phần tử của mảng 2 chiều  }  }  101 Mảng 2 chiều cần mấy chỉ số để xác định duy nhất 1 phần tử?  - Cần 2 chỉ số để xác định 1 phần tử duy nhất  102 Viết code tính tổng các phần tử của ma trận số (ý tưởng)?  function sumMatrix(matrix) {  let sum = 0;  for (let i = 0; i < matrix.length; i++) {  for (let j = 0; j < matrix[i].length; j++) {  sum += matrix[i][j];  }  }  return sum;  }  103 Viết code tính tổng các ô trên cùng hàng của ma trận số (ý tưởng)?  function sumRowElements(matrix) {  let rowSums = [];  for (let i = 0; i < matrix.length; i++) {  let sum = 0;  for (let j = 0; j < matrix[i].length; j++) {  sum += matrix[i][j];  }  rowSums.push(sum);  }  return rowSums;  }  let matrix = [  [1, 2, 3],  [4, 5, 6],  [7, 8, 9]  ];  104 Viết code tính tổng các ô trên cùng cột của ma trận số (ý tưởng)?  function sumColumnElements(matrix) {  let colSums = [];  for (let j = 0; j < matrix[0].length; j++) {  let sum = 0;  for (let i = 0; i < matrix.length; i++) {  sum += matrix[i][j];  }  colSums.push(sum);  }  return colSums;  }  let matrix = [  [1, 2, 3],  [4, 5, 6],  [7, 8, 9]  ];  105 Hàm (function) là gì?  - Là một nhóm các câu lệnh thực thi một nhiệm vụ nhất định  - Hàm dùng để nhóm các câu lệnh thành một khối có chức năng riêng biệt, để dễ quản lý và tái sử dụng mã.  - Các thành phần của hàm :   1. +Tên hàm: Đây là tên duy nhất của hàm, được sử dụng để gọi hàm từ nơi khác trong chương trình. 2. +Tham số (Parameters): Đây là các biến được đặt trong dấu ngoặc đơn sau tên hàm và dùng để truyền giá trị vào hàm khi gọi nó. 3. +Khối mã (Code Block): Đây là một tập hợp các câu lệnh, đoạn mã được thực thi khi hàm được gọi. Nó thực hiện các hành động hoặc tính toán và có thể sử dụng các tham số và biến khác. 4. +Giá trị trả về (Return Value): Hàm có thể trả về một giá trị sau khi thực hiện xong công việc. Nếu không có giá trị trả về được xác định, hàm sẽ trả về giá trị "undefined". |

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

- Tham số hình thức

+ Là các biến được định nghĩa trong phàn khai báo hàm

+ Nằm sau tên hàm, trong dấu ngoặc tròn. Dùng để đại diện cho các giá trị được truyền vào khi

hàm được gọi

+ Được sử dụng để thực hiện tính toán trong phần thân hàm

- Tham số thực

+ Là giá trị được truyền vào mỗi khi hàm được gọi

+ Nằm trong dấu ngoặc tròn sau tên hàm sau khi bạn gọi hàm

+ Giá trị của các tham số thực sẽ được gán cho các tham số hình thức tương ứng của hàm khi nó được thực thi

+ Số lượng và kiểu của tham số thực phải khớp với số lượng và kiểu của tham số hình thức được định nghĩa trong hàm.

107

108

- Hàm gọi (Calling Function):

+ Hàm gọi là hàm mà bạn sử dụng để gọi một hàm khác.

+ Trong một chương trình, khi bạn gọi một hàm, bạn sử dụng tên của hàm đó kèm theo các tham số thực (actual arguments) nếu có, và nó được bao quanh bởi cặp dấu ngoặc đơn ().

+ Hàm gọi chứa lời gọi đến hàm được gọi để thực hiện một tác vụ cụ thể hoặc tính toán một giá trị.

- Hàm được gọi (Called Function):

+ Hàm được gọi là hàm mà bạn gọi thông qua hàm gọi (calling function).

+ Khi một hàm được gọi, quá trình thực thi của chương trình chuyển sang hàm đó.

+ Hàm được gọi thực hiện các công việc được định nghĩa trong phần khối mã của nó và trả về kết quả nếu có.

110 Câu lệnh return dùng để làm gì?

- Câu lệnh "return" trong lập trình được sử dụng để trả về một giá trị từ một hàm. Khi hàm được gọi và gặp câu lệnh "return", nó sẽ dừng việc thực thi và trả về giá trị được chỉ định trong câu lệnh "return". Giá trị này có thể được sử dụng trong các biểu thức hoặc gán cho một biến.

111 - Có được phép viết “return;”?

Có, bạn hoàn toàn có thể viết câu lệnh "return;" trong một hàm mà không cần chỉ định giá trị trả về. Khi bạn sử dụng "return;" mà không có giá trị đi kèm, hàm sẽ trả về giá trị "undefined" mặc định.

- Khi bạn muốn kết thúc hàm mà không cần trả về giá trị nào cụ thể, bạn có thể sử dụng "return;".

function showMessage(message) {

console.log(message); // In ra thông báo

return; // Kết thúc hàm mà không trả về giá trị

}

- Khi bạn muốn kiểm tra một điều kiện và không cần trả về giá trị, bạn có thể sử dụng "return;" để thoát khỏi hàm sau khi kiểm tra điều kiện.

function isPositive(num) {

if (num > 0) {

console.log("Positive number.");

return; // Kết thúc hàm mà không trả về giá trị

}

console.log("Not a positive number.");

}

isPositive(7); // In ra "Positive number.

116. Lập trình hướng đối tượng (OOP) tập trung vào việc tạo ra các đối tượng và các mối quan hệ giữa chúng, trong khi lập trình hướng thủ tục/chức năng (Procedural/Functional Programming) tập trung vào các hàm và thủ tục để thực hiện các tác vụ. Trong lập trình hướng đối tượng, các đối tượng được tạo ra để đại diện cho các thực thể trong thế giới thực, trong khi lập trình hướng thủ tục/chức năng tập trung vào các hàm và thủ tục để thực hiện các tác vụ cụ thể.

117. Trong OOP, một class là một mô hình để tạo ra các đối tượng, trong khi một object là một thực thể được tạo ra từ một class. Một class định nghĩa các thuộc tính và phương thức của một đối tượng, trong khi một object là một thực thể cụ thể được tạo ra từ một class và có các giá trị cụ thể cho các thuộc tính.

118. "Môn học" là một khái niệm trừu tượng, trong khi "Môn học JavaScript" là một khái niệm cụ thể, do đó "Môn học JavaScript" là một object.

119. Sơ đồ lớp là một biểu đồ mô tả các class, thuộc tính và mối quan hệ giữa chúng. Nó thường được sử dụng để trình bày cấu trúc của một hệ thống phần mềm hoặc một ứng dụng.

120. Ví dụ về sơ đồ lớp của lớp Học viên:

Class: Học viên

Thuộc tính:

- Tên: string

- Tuổi: int

- Điểm: float

Phương thức:

- Nhập thông tin học viên()

- Hiển thị thông tin học viên()

121. Trong lập trình hướng đối tượng, thuộc tính là các đặc tính của đối tượng, ví dụ như tên, tuổi, địa chỉ. Các từ loại thường được sử dụng để biểu diễn thuộc tính là các kiểu dữ liệu như string, int, float, boolean, v.v.

122. Trong lập trình hướng đối tượng, phương thức là các hành động mà đối tượng có thể thực hiện. Các từ loại thường được sử dụng để biểu diễn phương thức là các hàm hoặc phương thức.

123. Cú pháp định nghĩa một class trong JavaScript như sau:

class ClassName {

// Thuộc tính và phương thức của class

}

124. Cú pháp tạo một đối tượng từ một class trong JavaScript như sau:

let objectName = new ClassName();

125. Một số class có sẵn trong JavaScript bao gồm: Array, Date, Math, String, v.v.

126. Để truy cập đến một thuộc tính của đối tượng trong JavaScript, bạn có thể sử dụng cú pháp:

objectName.propertyName;

Ví dụ: để truy cậpđến thuộc tính "name" của đối tượng "person", bạn có thể sử dụng cú pháp:

person.name;

127. Để truy cập đến một phương thức của đối tượng trong JavaScript, bạn có thể sử dụng cú pháp:

objectName.methodName();

Ví dụ: để gọi phương thức "displayInfo" của đối tượng "person", bạn có thể sử dụng cú pháp:

person.displayInfo();

128. Từ khóa "this" trong JavaScript được sử dụng để truy cập đến đối tượng hiện tại. Nó thường được sử dụng trong phương thức của một class để tham chiếu đến các thuộc tính và phương thức của đối tượng hiện tại.

129. DOM là viết tắt của Document Object Model, là một cách để truy cập và thay đổi các phần tử trên trang web bằng JavaScript. Nó cung cấp một cấu trúc phân cấp của các phần tử trên trang web, và cho phép JavaScript thao tác với chúng.

130. Thẻ <canvas> trong HTML được sử dụng để tạo ra một khu vực vẽ trên trang web, nơi mà bạn có thể sử dụng JavaScript để vẽ các hình ảnh và đồ họa.

131. Để vẽ một hình chữ nhật trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript như sau:

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

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

ctx.fillRect(x, y, width, height);

Trong đó "x" và "y" là tọa độ của góc trên bên trái của hình chữ nhật, "width" và "height" là kích thước của hình chữ nhật.

132. Để vẽ một hình tròn trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript như sau:

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

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

ctx.beginPath();

ctx.arc(x, y, radius, startAngle, endAngle);

ctx.fill();

Trong đó "x" và "y" là tọa độ của tâm của hình tròn, "radius" là bán kính của hình tròn, "startAngle" và "endAngle" là góc bắt đầu và kết thúc của hình tròn.

133. Để vẽ text trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript như sau:

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

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

ctx.font = "30px Arial";

ctx.fillText("Hello, world!", x, y);

Trong đó "font" là kiểu font chữ và kích thước, "fillText()" là phương thức để vẽ text, "x" và "y" là tọa độ của văn bản.

134. Để di chuyển một hình tròn trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript để xóa hình tròn cũ và vẽ hình tròn mới với các tọa độ mới. Bạn có thể sử dụng các phương thức như "clearRect()" để xóa hình tròn cũ và "fillCircle()" để vẽ hình tròn mới với các tọa độ mới.

116. Lập trình hướng đối tượng (OOP) tập trung vào việc tạo ra các đối tượng và các mối quan hệ giữa chúng, trong khi lập trình hướng thủ tục/chức năng (Procedural/Functional Programming) tập trung vào các hàm và thủ tục để thực hiện các tác vụ. Trong lập trình hướng đối tượng, các đối tượng được tạo ra để đại diện cho các thực thể trong thế giới thực, trong khi lập trình hướng thủ tục/chức năng tập trung vào các hàm và thủ tục để thực hiện các tác vụ cụ thể.

117. Trong OOP, một class là một mô hình để tạo ra các đối tượng, trong khi một object là một thực thể được tạo ra từ một class. Một class định nghĩa các thuộc tính và phương thức của một đối tượng, trong khi một object là một thực thể cụ thể được tạo ra từ một class và có các giá trị cụ thể cho các thuộc tính.

118. "Môn học" là một khái niệm trừu tượng, trong khi "Môn học JavaScript" là một khái niệm cụ thể, do đó "Môn học JavaScript" là một object.

119. Sơ đồ lớp là một biểu đồ mô tả các class, thuộc tính và mối quan hệ giữa chúng. Nó thường được sử dụng để trình bày cấu trúc của một hệ thống phần mềm hoặc một ứng dụng.

120. Ví dụ về sơ đồ lớp của lớp Học viên:

Class: Học viên

Thuộc tính:

- Tên: string

- Tuổi: int

- Điểm: float

Phương thức:

- Nhập thông tin học viên()

- Hiển thị thông tin học viên()

121. Trong lập trình hướng đối tượng, thuộc tính là các đặc tính của đối tượng, ví dụ như tên, tuổi, địa chỉ. Các từ loại thường được sử dụng để biểu diễn thuộc tính là các kiểu dữ liệu như string, int, float, boolean, v.v.

122. Trong lập trình hướng đối tượng, phương thức là các hành động mà đối tượng có thể thực hiện. Các từ loại thường được sử dụng để biểu diễn phương thức là các hàm hoặc phương thức.

123. Cú pháp định nghĩa một class trong JavaScript như sau:

class ClassName {

// Thuộc tính và phương thức của class

}

124. Cú pháp tạo một đối tượng từ một class trong JavaScript như sau:

let objectName = new ClassName();

125. Một số class có sẵn trong JavaScript bao gồm: Array, Date, Math, String, v.v.

126. Để truy cập đến một thuộc tính của đối tượng trong JavaScript, bạn có thể sử dụng cú pháp:

objectName.propertyName;

Ví dụ: để truy cậpđến thuộc tính "name" của đối tượng "person", bạn có thể sử dụng cú pháp:

person.name;

127. Để truy cập đến một phương thức của đối tượng trong JavaScript, bạn có thể sử dụng cú pháp:

objectName.methodName();

Ví dụ: để gọi phương thức "displayInfo" của đối tượng "person", bạn có thể sử dụng cú pháp:

person.displayInfo();

128. Từ khóa "this" trong JavaScript được sử dụng để truy cập đến đối tượng hiện tại. Nó thường được sử dụng trong phương thức của một class để tham chiếu đến các thuộc tính và phương thức của đối tượng hiện tại.

129. DOM là viết tắt của Document Object Model, là một cách để truy cập và thay đổi các phần tử trên trang web bằng JavaScript. Nó cung cấp một cấu trúc phân cấp của các phần tử trên trang web, và cho phép JavaScript thao tác với chúng.

130. Thẻ <canvas> trong HTML được sử dụng để tạo ra một khu vực vẽ trên trang web, nơi mà bạn có thể sử dụng JavaScript để vẽ các hình ảnh và đồ họa.

131. Để vẽ một hình chữ nhật trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript như sau:

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

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

ctx.fillRect(x, y, width, height);

Trong đó "x" và "y" là tọa độ của góc trên bên trái của hình chữ nhật, "width" và "height" là kích thước của hình chữ nhật.

132. Để vẽ một hình tròn trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript như sau:

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

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

ctx.beginPath();

ctx.arc(x, y, radius, startAngle, endAngle);

ctx.fill();

Trong đó "x" và "y" là tọa độ của tâm của hình tròn, "radius" là bán kính của hình tròn, "startAngle" và "endAngle" là góc bắt đầu và kết thúc của hình tròn.

133. Để vẽ text trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript như sau:

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

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

ctx.font = "30px Arial";

ctx.fillText("Hello, world!", x, y);

Trong đó "font" là kiểu font chữ và kích thước, "fillText()" là phương thức để vẽ text, "x" và "y" là tọa độ của văn bản.

134. Để di chuyển một hình tròn trên canvas, bạn có thể sử dụng các phương thức của đối tượng "CanvasRenderingContext2D" trong JavaScript để xóa hình tròn cũ và vẽ hình tròn mới với các tọa độ mới. Bạn có thể sử dụng các phương thức như "clearRect()" để xóa hình tròn cũ và "fillCircle()" để vẽ hình tròn mới với các tọa độ mới.