**CÂU HỎI AUDIT MODULE 1**

**BOOTCAMP PREPARATION 2.0**

1. Repository là nơi lưu trữ mã nguồn, tài liệu, dữ liệu  
   Có hai loại repository:  
   + Local repository  
   + Remote repository
2. git init: Khởi tạo một kho chứa Git mới trong thư mục hiện tại. Dùng lệnh này khi bạn muốn bắt đầu theo dõi các thay đổi trong thư mục này.

git clone [URL]: Sao chép một kho chứa Git từ URL (đường dẫn) đã cung cấp vào máy tính của bạn.

git add [file(s)]: Thêm các tập tin vào chỉ mục (staging area), chuẩn bị cho việc commit. Ví dụ: git add filename hoặc git add . để thêm tất cả các tập tin thay đổi.

git commit -m "[message]": Tạo một commit với những thay đổi đã được thêm vào chỉ mục. Thông điệp commit giúp bạn diễn đạt tóm tắt về những gì đã thay đổi trong commit này.

git status: Hiển thị trạng thái hiện tại của kho chứa, bao gồm các tập tin đã thay đổi và trạng thái của chúng.

git log: Hiển thị lịch sử các commit trên nhánh hiện tại.

git pull: Kéo (pull) các thay đổi từ kho chứa từ xa (remote repository) về máy tính của bạn, hợp nhất chúng vào nhánh hiện tại.

git push: Đẩy (push) các thay đổi từ máy tính của bạn lên kho chứa từ xa.

1. Giải thuật là một tập hợp các bước cụ thể và có thứ tự được sắp xếp để giải quyết một vấn đề hoặc thực hiện một tác vụ.  
   Có 3 cách biểu diễn thuật toán: lưu đồ, mã giả và ngôn ngữ lập trình
2. TìmKiếm(arr, x):  
   Duyệt qua từng vị trí i từ 0 đến chiều dài của arr:  
   Nếu arr[i] bằng x, trả về i.  
   Trả về -1.
3. Thuật toán Sắp Xếp Mảng Số Nguyên (Bubble Sort):

Input: Mảng arr chứa các số nguyên.

Output: Mảng arr sau khi đã được sắp xếp.

Bước 1: Lặp qua từng phần tử i từ 0 đến chiều dài của arr - 1:

Bước 1.1: Lặp qua từng phần tử j từ 0 đến chiều dài của arr - i - 1:

Nếu arr[j] > arr[j+1]:

Hoán đổi giá trị của arr[j] và arr[j+1].

Kết thúc thuật toán.

1. Thuật toán: Tìm số lớn nhất trong mảng

Ý tưởng:

Thuật toán Tìm số lớn nhất trong mảng tìm kiếm qua từng phần tử trong mảng và lưu giữ số lớn nhất mà nó đã thấy cho đến nay.

1. Danh sách không có thứ tự (<ul> - Unordered List):

Sử dụng <ul> để bắt đầu danh sách không có thứ tự.

Sử dụng <li> cho từng mục trong danh sách.

Danh sách có thứ tự (<ol> - Ordered List):

Sử dụng <ol> để bắt đầu danh sách có thứ tự.

Sử dụng <li> cho từng mục trong danh sách.

1. Thẻ <table>: Thẻ này tạo ra cấu trúc chính của bảng, bao bọc toàn bộ bảng.

Thẻ <tr> (Table Row): Thẻ này tạo ra một hàng trong bảng. Bạn sử dụng thẻ <tr> để tạo các hàng riêng lẻ của bảng.

Thẻ <th> (Table Header Cell): Thẻ này tạo ra ô tiêu đề trong bảng. Các ô này thường được sử dụng để đặt tiêu đề cho cột hoặc hàng. Các ô tiêu đề thường được làm nổi bật hơn so với các ô thông thường, như in đậm hoặc có màu nền khác biệt.

Thẻ <td> (Table Data Cell): Thẻ này tạo ra ô dữ liệu trong bảng. Các ô này chứa thông tin thực sự trong bảng, chẳng hạn như dữ liệu cột và hàng.

1. innerHTML: Thuộc tính để thao tác nội dung HTML bên trong phần tử. Có thể thêm/đổi thẻ, thuộc tính và nội dung HTML mới. Xử lý các thẻ và thuộc tính.

innerText: Thuộc tính để thao tác với nội dung văn bản hiển thị cho người dùng. Không xử lý HTML, chỉ là văn bản thô.

1. <html>: Đánh dấu đầu vùng chứa toàn bộ nội dung HTML.

<head>: Chứa thông tin không hiển thị trực tiếp trên trình duyệt, chẳng hạn như thẻ <title>.

<body>: Chứa nội dung hiển thị trực tiếp trên trình duyệt.

<h1>, <h2>, <h3>, ..., <h6>: Đánh dấu tiêu đề theo cấp độ 1 đến 6.

<p>: Đánh dấu đoạn văn bản.

<a>: Tạo liên kết (hyperlink) đến trang web khác.

<img>: Hiển thị hình ảnh.

<ul>: Tạo danh sách không có thứ tự.

<ol>: Tạo danh sách có thứ tự.

<li>: Đánh dấu từng mục trong danh sách.

<table>: Tạo bảng.

<tr>: Đánh dấu hàng trong bảng.

<th>: Đánh dấu ô tiêu đề trong bảng.

<td>: Đánh dấu ô dữ liệu trong bảng.

<div>: Tạo khối chứa phần tử và CSS.

<span>: Tạo phần tử văn bản nhỏ, có thể được kiểm soát bằng CSS.

<form>: Tạo biểu mẫu nhập liệu.

<input>: Tạo ô nhập liệu.

<textarea>: Tạo ô văn bản nhập liệu dài.

<button>: Tạo nút bấm.

<br>: Tạo dòng mới.

<hr>: Tạo đường ngang ngăn cách phần tử.

1. action: Xác định URL hoặc đường dẫn tới trang xử lý dữ liệu khi biểu mẫu được gửi đi.

method: Xác định phương thức HTTP được sử dụng để gửi dữ liệu. Có hai giá trị thường dùng là "GET" và "POST".

name: Đặt tên cho biểu mẫu để tham chiếu bằng mã JavaScript hoặc CSS.

1. Phương thức **GET**:

Dữ liệu gửi đi qua URL: Dữ liệu biểu mẫu được gắn vào URL và hiển thị trực tiếp trên thanh địa chỉ của trình duyệt.

Số lượng dữ liệu giới hạn: Do dữ liệu được đính kèm vào URL, có giới hạn về kích thước dữ liệu gửi đi. Trình duyệt và máy chủ cũng có thể giới hạn.

Phù hợp cho dữ liệu nhỏ: Phương thức GET thường được sử dụng cho các tìm kiếm, lọc, hoặc yêu cầu dữ liệu nhỏ gọn từ máy chủ.

Dễ theo dõi và chia sẻ: Vì dữ liệu hiển thị trên URL, dễ dàng sao chép và dán vào trình duyệt hoặc chia sẻ.

Phương thức **POST**:

Dữ liệu gửi đi trong phần thân yêu cầu: Dữ liệu biểu mẫu được gửi trong phần thân yêu cầu HTTP và không hiển thị trên URL.

Số lượng dữ liệu lớn hơn: Vì dữ liệu không bị giới hạn bởi URL, phương thức POST thích hợp cho dữ liệu lớn hơn, chẳng hạn như tải tệp lên.

An toàn hơn: Vì dữ liệu không hiển thị trên URL, nó an toàn hơn đối với dữ liệu nhạy cảm hoặc mật khẩu.

Phù hợp cho dữ liệu lớn hoặc nhạy cảm: Phương thức POST thường được sử dụng khi gửi dữ liệu lớn, nhạy cảm hoặc thay đổi dữ liệu trên máy chủ.

Tóm lại, phương thức GET thích hợp cho dữ liệu nhỏ và khi bạn muốn dễ dàng chia sẻ dữ liệu, trong khi phương thức POST thích hợp cho dữ liệu lớn, nhạy cảm hoặc cần thay đổi dữ liệu trên máy chủ.

1. let: Dùng để khai báo biến với phạm vi giới hạn trong khối mã, không bị hoisting, và không cho phép khai báo lại biến trong cùng một phạm vi.

var: Dùng để khai báo biến với phạm vi toàn bộ hàm, bị hoisting, và cho phép khai báo lại biến trong cùng một phạm vi.

1. Number: Kiểu số, bao gồm cả số nguyên và số thập phân.

String: Kiểu chuỗi, là văn bản được đặt trong dấu ngoặc đơn hoặc ngoặc kép.

Boolean: Kiểu logic, chỉ có hai giá trị là true và false.

Undefined: Biến chưa được khai báo hoặc không có giá trị.

Null: Biến có giá trị null, tức là không có giá trị.

Object: Kiểu đối tượng, gồm các cặp key-value.

Symbol: Kiểu ký hiệu, được sử dụng để tạo ra các giá trị không thể thay đổi duy nhất.

1. Sử dụng dấu escape (\):  
   Sử dụng dấu ngoặc kép với dấu nháy đơn bên trong và ngược lại.
2. NaN là giá trị đặc biệt trong JavaScript, biểu thị cho "Not-a-Number" (không phải là số). Nó xuất hiện khi có phép toán số học không hợp lệ hoặc chuyển đổi giá trị thành số không thành công.

NaN === NaN trả về false, vì NaN không bao giờ bằng chính nó hoặc bất kỳ giá trị khác.

1. let x = 5;

let a = ++x;

console.log(a); // Output: 6

console.log(x); // Output: 6

let x = 5;

let b = x++;

console.log(b); // Output: 5

console.log(x); // Output: 6

1. Toán tử số học (+, -, \*, /, %)

Toán tử so sánh (<, >, <=, >=)

Toán tử so sánh bằng (==, ===, !=, !==)

Toán tử logic (&&, ||, !)

Toán tử gán (=, +=, -=, \*=, /=, %=)

Toán tử tăng và giảm (++x, x++, --x, x--)

Toán tử điều kiện (?:)

Toán tử chuỗi (+)

1. Toán tử ba ngôi, còn được gọi là toán tử điều kiện (ternary operator), là một loại toán tử trong JavaScript cho phép bạn thực hiện một phép so sánh và trả về một giá trị dựa trên kết quả của phép so sánh đó. Cú pháp:

condition ? expressionIfTrue : expressionIfFalse;

1. null: Sự thiếu vắng hoặc không tồn tại giá trị.

"Rỗng" (empty): Sự tồn tại nhưng không có giá trị hoặc phần tử.

1. Giống nhau: đều để so sánh bằng hai biểu thức  
   Khác nhau: == : so sánh hai biểu thức mà không quan tâm đến kiểu dữ liệu  
   === : cũng so sánh hai biểu thức nhưng đồng thời so sánh kiểu dữ liệu
2. && (Toán tử AND):

Toán tử && trả về true nếu cả hai biểu thức hoặc điều kiện đều đúng (true).

Nếu một trong hai biểu thức hoặc điều kiện là false, thì kết quả là false.

|| (Toán tử OR):

Toán tử || trả về true nếu ít nhất một trong hai biểu thức hoặc điều kiện là đúng (true).

Chỉ khi cả hai biểu thức hoặc điều kiện đều là false, kết quả mới là false.

1. Các hàm phổ biến nhất để thực hiện công việc này là alert(), confirm(), và prompt().
2. Có ba câu lệnh điều khiển:  
   +Câu lệnh điều kiện (if, if else, switch case)  
   +Câu lệnh lặp (for, while, do while)  
   +Câu lệnh nhảy (break, continue, return)
3. If : có thể so sánh lớn hơn, nhỏ hơn hoặc bằng; chỉ có một khối lệnh thực thi nếu điều kiện đúng; mỗi câu lệnh if có một biểu thức điều kiện với giá trị trả về là Boolean  
   Switch case: chỉ so sánh bằng hoặc khác nhau; tất cả case đều so sánh với biểu thức ban đầu; giá trị trả về là number hoặc string; nếu điều kiện đúng mà không có câu lệnh break thì tất cả các case dưới đều được thực hiện.
4. switch case so sánh giá trị bằng === (so sánh nghiêm ngặt).

if bậc thang có thể sử dụng == hoặc === tùy thuộc vào cách bạn thiết lập điều kiện.

Sử dụng switch case khi có nhiều trường hợp cụ thể và bạn muốn tối ưu hóa mã.

Sử dụng if bậc thang khi cần kiểm tra điều kiện linh hoạt hơn hoặc muốn quản lý chuyển đổi kiểu tự động.

1. Biểu thức khởi tạo (Initialization Expression): Biểu thức này thường được sử dụng để khởi tạo một biến đếm hoặc thực hiện một công việc ban đầu trước khi vòng lặp bắt đầu. Biểu thức này không bắt buộc. Nếu thiếu, không có gì được thực hiện trước vòng lặp.

Biểu thức điều kiện (Condition Expression): Biểu thức này đánh giá sau mỗi lần lặp, và nếu nó trả về true, vòng lặp tiếp tục chạy; nếu trả về false, vòng lặp kết thúc. Biểu thức này bắt buộc. Nếu thiếu và bạn không có một điểm dừng nào, vòng lặp có thể chạy vô hạn.

Biểu thức tăng/giảm (Increment/Decrement Expression): Biểu thức này thường được sử dụng để tăng hoặc giảm giá trị biến đếm sau mỗi lần lặp. Biểu thức này không bắt buộc. Nếu thiếu, bạn phải đảm bảo có một cách khác để thay đổi giá trị kiểm tra biểu thức điều kiện, hoặc có nguy cơ vòng lặp lặp vô hạn.

1. Hãy xem xét một bài toán đơn giản: "Viết một chương trình để tính tổng của tất cả các số nguyên từ 1 đến n."

Bài toán này yêu cầu tính tổng của một dãy số liên tiếp từ 1 đến n, trong đó n là một số nguyên dương cho trước. Điều này đòi hỏi lặp qua từng số từ 1 đến n và cộng chúng lại. Trong trường hợp này, bạn có thể sử dụng vòng lặp for để giải quyết bài toán:

1. Giống nhau: để dung để lặp  
   Khác nhau:   
   for: Dùng khi biết trước số lần lặp cụ thể.

while: Dùng khi số lần lặp không biết trước hoặc phụ thuộc vào điều kiện.

do...while: Dùng khi cần đảm bảo ít nhất một lần lặp và số lần lặp không biết trước.

1. break được sử dụng để kết thúc vòng lặp ngay lập tức khi một điều kiện đúng.

continue được sử dụng để bỏ qua phần còn lại của lần lặp hiện tại và chuyển đến lần lặp tiếp theo.

1. Mảng một chiều trong JavaScript là một tập hợp tuần tự của các phần tử được lưu trữ dưới dạng một biến đối tượng. Chúng có chỉ số bắt đầu từ 0, có thể chứa các loại dữ liệu khác nhau, có khả năng thay đổi độ dài và cung cấp nhiều phương thức và thuộc tính để làm việc với dữ liệu mảng.
2. Sử dụng cú pháp [].

Sử dụng hàm khởi tạo new Array() và truyền các giá trị chuỗi.

Sử dụng hàm khởi tạo Array() và truyền các giá trị chuỗi.

1. push(): Thêm phần tử vào cuối mảng.

pop(): Xóa và trả về phần tử cuối cùng của mảng.

unshift(): Thêm phần tử vào đầu mảng.

shift(): Xóa và trả về phần tử đầu tiên của mảng.

1. Tham trị (Pass by Value) áp dụng cho kiểu dữ liệu nguyên thủy. Gán một biến cho biến khác sao chép giá trị.

Tham chiếu (Pass by Reference) áp dụng cho đối tượng và mảng. Gán một biến cho biến khác tạo ra tham chiếu đến cùng một vị trí bộ nhớ. Thay đổi qua một biến ảnh hưởng đến tất cả các biến khác tham chiếu đến nó.

1. Hàm có return trả về giá trị cụ thể sau khi thực thi.

Hàm không có return thực hiện hành động mà không trả về giá trị cụ thể.

1. Lập trình hướng đối tượng (OOP) là một phương pháp lập trình dựa trên các đối tượng có dữ liệu và hành vi.
2. Class: Là bản thiết kế hoặc mô tả của đối tượng, định nghĩa thuộc tính và phương thức.

Object (Đối tượng): Là một thể hiện cụ thể của lớp, có dữ liệu và hành vi riêng biệt.

1. **Encapsulation (Đóng gói)**:

Encapsulation là tính chất cho phép che giấu (private) dữ liệu của một đối tượng và chỉ cho phép truy cập đến nó thông qua các phương thức công khai (public methods).

Ví dụ: Một lớp "BankAccount" có thuộc tính "số dư" được che giấu và có các phương thức "nạp tiền" và "rút tiền" để truy cập và cập nhật số dư.

**Inheritance (Kế thừa)**:

Inheritance cho phép tạo ra các lớp mới (lớp con) dựa trên các lớp đã có sẵn (lớp cha). Lớp con kế thừa các thuộc tính và phương thức của lớp cha.

Ví dụ: Lớp cha "Vehicle" có thuộc tính "số bánh xe," và lớp con "Car" kế thừa thuộc tính này, nhưng có thể có thêm các thuộc tính riêng như "màu sắc."

**Polymorphism (Đa hình)**:

Polymorphism cho phép các đối tượng của các lớp khác nhau thực hiện cùng một phương thức theo cách riêng biệt. Điều này giúp giảm thiểu sự phụ thuộc vào kiểu dữ liệu cụ thể.

Ví dụ: Cả lớp "Circle" và "Rectangle" có phương thức "tính diện tích," nhưng mỗi lớp thực hiện nó theo cách khác để tính diện tích của hình tròn và hình chữ nhật.

**Abstraction (Trừu tượng)**:

Abstraction là tính chất cho phép ẩn đi các chi tiết không quan trọng và tập trung vào các khái niệm quan trọng. Điều này giúp đơn giản hóa việc phát triển và duy trì mã nguồn.

Ví dụ: Lớp "Animal" có phương thức "speak," nhưng không quan tâm đến cách cụ thể từng loài động vật kêu. Các lớp con như "Dog" và "Cat" sẽ cung cấp triển khai cụ thể cho "speak."

1. Constructor trong lập trình hướng đối tượng là một phương thức đặc biệt trong một lớp được sử dụng để khởi tạo đối tượng của lớp đó. Trong một lớp, có thể có nhiều constructor, nhưng chúng phải có danh sách tham số khác nhau (overloading).
2. length: Trả về độ dài của chuỗi.

slice(start, end): Tương tự substring và hỗ trợ index âm.

indexOf(searchValue, startIndex): Tìm vị trí đầu tiên của searchValue trong chuỗi.

1. var canvas = document.getElementById("myCanvas"); //Lấy tham chiếu  
   var context = canvas.getContext("2d"); //Lấy 1 ngữ cảnh  
   context.beginPath(); // Bắt đầu một đường vẽ mới  
   context.arc(150, 150, 50, 0, 2 \* Math.PI); //tọa độ x,y;bán kính;góc bắt đầu; góc kết thúc.  
   context.stroke(); // Vẽ đường viền