AUDIT MODULE 1

|  |  |  |
| --- | --- | --- |
| stt | Câu hỏi | Trả lời |
| 1 | Repository là gì? có bao nhiêu loại repository? | - Repository (repo): là nơi chứa toàn bộ mã nguồn , bao gồm các file và lịch sử của file đó , chứa tất cả các commit  - Có 2 loại :   * Local repository : trên máy của clever (LTV) * Remote repository : trên máy chủ được chia sẻ (github) |
| 2 | Các cấu lệnh căn bản để làm việc với git? | - git clone : sao chép 1 reponsitory  - git init : khởi tạo 1 repository (Local repo)  - git add : đưa các file vào trong vùng staged  - git commit : ghi nhận các thay đổi  - git push : đưa ra các thay đổi từ local repository lên remote repository  - git pull : remote -> local |
| 3 | Giải thuật là gì? Các cách để biểu diễn thuật toán? | - Giải thuật / thuật toán : là tập hợp hữu hạn các bước để giải quyết 1 vấn đề .  - Có nhiều cách khác nhau để mô tả thuật toán , thông dụng là : Mã giải (Pseudo-code), Lưu đồ (Flowchart),Ngôn ngữ lập trình |
| 4 | Mô tả thuật toán tìm kiếm 1 phần tử trong mảng bằng mã giả? |  |
| 5 | Mô tả thuật toán sắp xếp mảng số nguyên bằng mã giả? |  |
| 6 | Trình bày ý tưởng của 1 thuật toán bất kì? |  |
| 7 | Nêu các thẻ để tạo danh sách ?  Sự khác nhau giữa <ul> và <ol> ? | Các thẻ để tạo danh sách  1.Thẻ <ul> để định nghĩa một danh sách không xếp thứ tự  2.Thuộc tính CSS list-style-type để định nghĩa biểu tượng cho các phần tử trong danh sách  3.Thẻ <ol> để định nghĩa một danh sách xếp thứ tự  4.Thuộc tính type để định nghĩa kiểu đánh thứ tự  5.Thẻ <li> để định nghĩa một phần tử trong danh sách  6.Thẻ <dl> để định nghĩa một danh sách miêu tả  7.Thẻ <dt> để định nghĩa một thuật ngữ  8.Thẻ <dd> để miêu tả thuật ngữ ngay phía trên nó trong danh sách miêu tả  Sự khác nhau giữa <ul> và <ol>  - Các danh sách không có thứ tự kiểu như thế này: Được tạo thành từ các thẻ ul và li  - Danh sách được đánh thứ tự kiểu như thế này: Được tạo thành từ các thẻ ol và li. |
| 8 | Các thẻ tạo nên 1 table? Phân biệt th và td? | - Các thẻ được dùng để tạo bảng :  <table> : Khai báo 1 bảng  <tr> : tạo 1 dòng  <td> : tạo 1 ô DL  <th> : khai báo tiêu đề  - Phân biệt td và th  Td : mặc định viết thường và căn lề trái  Th : mặc định viết in đậm và căn giữa |
| 9 | Phân biệt innerHTML và innerText? | innerHTML sẽ in ra nội dung text và những thẻ trong div này luôn.  innerText chir ra nd text |
| 10 | Nêu một số thẻ HTML cơ bản mà bạn biết | - Thẻ tiêu đề <th>  - Thẻ tạo đoạn văn bản <p>  - Thẻ tạo liên kết <href >  - Thẻ chèn hình ảnh <img>  - Thẻ tạo danh sách <ul> , <ol> , <li>  - Thẻ chèn audio  - Thẻ chèn video |
| 11 | Các thuộc tính cơ bản của thẻ form | - Thuộc tính “action” : có giá trị là 1 URL , quy định nơi dữ liệu được gởi đến  - Thuộc tính “name” : quy định tên của form , có thể dùng để truy xuất đến các giá trị trong form  - Thuộc tính “method” : quyết định phương thức gởi đi . Bao gồm GET và POST . |
| 12 | Phân biệt phương thức get và post trong thẻ form. | GET (lấy DL về )  - DL được gởi đi **sẽ hiển thị** trên thanh địa chỉ của trình duyệt .  - Không nên sử dụng để gởi những DL quan trọng  - Thường được sd để gởi những DL nhỏ  - Chỉ giử được dạng TEXT .  (Thanh URL sẽ giới hạn dung lượng nên không sd GET)  POST (gửi DL lên)  - DL được gởi k hiển thị trên thanh địa chỉ của trình duyệt  - Được sd để gởi các DL quan trọng  - K hạn chế dung lượng gửi đi  - Thường được gửi những DL lớn lên server  - Gửi được âm thanh , video , siêu văn bản , … |
| 13 | Cách khai báo biến bằng từ khóa let và var khác nhau như thế nào ?. Phạm vi của biến? | LET (khai báo 1 biến tạm)  - Không cho phép khai báo trùng tên biến  - Không cho phép sd trước khi khai báo  - Chỉ được truy cập trong phạm vi Block Scope  VAR  - Cho phép khai báo cùng tên biến  - Cho phép sd trước khi khai báo (vì có cơ chế hosting : các biến được khai báo bằng VAR sẽ được đưa lên đầu chương trình khi chạy)  - Truy cập trong phạm vi Function Scope  (Khi khai báo DL var sẽ lấy biến khai báo sau đè lên biến được khai báo trước đó)  CONST (gán giá trị)  - Sử dụng để khai báo hằng số  - Tính chất sd giống LET  (Khi sd CONST thì bắt buộc phải gán giá trị cho biến và không thể thay đổi vùng nhớ) |
| 14 | Trong JavaScript có bao nhiêu loại kiểu dữ liệu ?. Làm thế nào để xác định được biến có kiểu dữ liệu gì? | - Kiểu DL có 2 loại : kiểu Nguyên Thuỷ & kiểu Đối Tượng  Kiểu nguyên thuỷ : String , Number , Boolean , Null (Kiểu DL : object) , Underfined  Kiểu đối tượng : Object , Array , Function , Sysbol (Kiểu DL : unique) -> ES6  - Dùng TYPE OF : để xđ kiểu DL của 1 biến |
| 15 | Các cách tạo chuỗi chứa dấu nháy ? | Khai báo bên trong dấu “nháy đơn” hoặc dấu “nháy kép” |
| 16 | NaN là gì? NaN === NaN có đúng không? Vì sao? | NaN : Not a Number |
| 17 | Cho 1 bài toán liên quan đến ++ trước và ++ sau. Yêu cầu dự đoán kết quả, và tại sao lại có kết quả như vậy? | Ví dụ :  var x; // Khai báo biến x    x = 10;  alert(++x); // Outputs: 11  alert(x); // Outputs: 11    x = 10;  alert(x++); // Outputs: 10  alert(x); // Outputs: 11    x = 10;  alert(--x); // Outputs: 9  alert(x); // Outputs: 9    x = 10;  alert(x--); // Outputs: 10  alert(x); // Outputs: 9  - **++ trước** sẽ tăng giá trị lên 1 đơn vị rồi mới đánh giá biểu thức  - **++ sau** sẽ đánh giá biển thức trước rồi mới tăng hoặc giảm giá trị 1 đơn vị |
| 18 | Các loại toán tử trong js? Cho biết độ ưu tiên của các toán tử trong một biểu thức? | Các loại toán tử : toán tử số học , so sánh , logic , gán , cộng chuỗi .  Độ ưu tiên : Số học > So sánh > Logic > Gán |
| 19 | Toán tử ba ngôi là gì? Cú pháp? | Cú pháp :  Biểu thức điều kện ? Biểu thức ĐÚNG : Biểu thức SAI |
| 20 | So sánh null và rỗng |  |
| 21 | Toán tử == và === khác và giống nhau như thế nào | - == : So sánh giá trị hình thức  - === : So sánh giá trị hình thức và kiểu DL |
| 22 | Phân biệt giữa 2 toán tử && và || trong JavaScript? | - && : Tất cả các vế đều là đúng (T – T : F - F)  - || : Có thể là 1 vế đúng , 1 vế sai |
| 23 | Các hàm dùng để tạo thông báo trong JavaScript ? | Có 3 hàm :  - Console.log : hiển thị thông báo trong Console  -Document.write : Xuất kết quả thông báo ra màng hình  -Alret : Hiển thị hộp thoại thông báo |
| 24 | Có bao nhiêu Statement control(câu lệnh điều khiển) trong Java Script ? | Có 3 loại : If , If – Else , Switch- Case  Câu lệnh If bao gồm :  If đơn  If đủ (if – else)  If bậc thang (if – else , if – else ,…)  If lồng nhau |
| 25 | So sánh sự khác nhau giữa if và switch case? | If  - Biểu thức đk trả về Boolean  - Sd với mọi loại so sánh  - Mỗi biểu thức đk đúng chỉ thực thi 1 khối lệnh If  - Mỗi khối lệnh If chỉ có 1 biểu thức đk  Switch – Case  - Trả về tất cả các kiểu DL nguyên thuỷ  - Sd = = = hoặc khác (default)  - Sẽ thực thi tất cả các case đến khi gặp Break hoặc dừng chương trình  - 1 biểu thức đk so sánh với tất cả các case  (Nếu như khối đk if có > hoặc = 4 biểu tức đk thì nên sd Switch - Case) |
| 26 | Switch case so sánh == hay ===. Đặt ra trường hợp là so sánh bằng thì khi nào sử dụng if bậc thang? Khi nào sử dụng switch case | - switch ... case so sánh === (so sánh giá trị + kiểu dữ liệu)  - So sánh bằng khi biểu thức có ít hơn 3 điều kiện thì sử dụng if bậc thang  - Sử dụng switch ... case khi có nhiều hơn 3 điều kiện |
| 27 | Các biểu thức và luồng thực thi của for. Nếu thiếu 1 hoặc tất cả các biểu thức thì vòng for sẽ chạy như thế nào? | -Cú pháp :  For (biểu thức khởi tạo (1) ; biểu thức điều kiện (2) ; biểu thức tăng hoặc giảm (3) {  // Thân vòng lặp (Loop Body) (4)  }  - Luồng thực thi : 1->2->4->3->2…Tới khi biểu thức 2 là “false”  - Có thể tách biến (1) & (3) ra khỏi vòng lặp for.  Đối với vòng lặp for có thể thiếu 1,2 or 3 biểu thức thì vòng lặp vẫn thực thi được . Tuy nhiên phải có đủ 2 dấu chấm phẩy “ ; ” |
| 28 | Đặt ra 1 bài toán. Xác định bài toán cần sử dụng vòng lặp nào? | Vòng lặp for...of : được ra mắt ở phiên bản ES6. Tương tự như for, ᴠòng lặp nàу được ѕử dụng để duуệt từng phần tử của đối tượng duуệt. Số lượng lặp bằng ᴠới ѕố phần tử của đối tượng. Vòng lặp for...in : thường được ѕử dụng ᴠới mục đích là lặp trong một ****object**** chứ không phải trong ****arraу**** haу ****ѕtring**** giống như hai ᴠòng lặp trên. Số lượng lặp tương ứng ᴠới ѕố thuộc tính của object mà ta duуệt |
| 29 | So sánh giống và khác nhau giữa for, while và do…while | FOR  - Sd khi biết trước số lần lặp  - Xét đk trước khi lặp  WHILE  - Không biết trước số lần lặp  - Xét đk sau khi lặp  DO – WHILE  - Không biết trước số lần lặp  - Xét đk sau khi lặp -> có thể thực thi 1 lần dù đk sai ngay từ đầu |
| 30 | So sánh break và continue | BREAK  - Khi vòng lặp gặp câu lệnh Break thì lập tức thoát khỏi vòng lặp .  CONTINUE  - Khi gặp câu lệnh continue thì sẽ bỏ qua các câu lệnh bên dưới comment & thực thi vòng lặp tiếp theo |
| 31 | Đặc điểm mảng một chiều trong JavaScript | - Là 1 biến đặc biệt có thể lưu trữ nhiều giá trị  - Giá trị trong mảng thì được gọi là phần tử  - Các phần tử được lưu trữ liền kề lẫn nhau , bắt đầu từ vị trí (index) 0 & kết thúc tại vị trí length -1 (length : là độ dài của mảng)  - Gọi mảng là biến đặc biệt vì có thể lưu trữ 1 lúc nhiều giá trị và nhiều kiểu DL khác nhau (đối với JS) |
| 32 | Các cách khởi tạo một mảng kiểu String trong JavaScript | Có 3 cách:  - Let array = [ ];  - Let arr = new Array ( );  - Let arr = Array ( ); |
| 33 | - 1 số hàm thao tác với mảng?  - Phân biệt push() và unshift()?  - Phân biệt push() và pop()?  - Phân biệt shift() và unshift()? | - Một số hàm có sẵn thao tác với mảng :  1. Push () : Thêm phần tử vào cuối mảng - trả về độ dài của mảng sau khi được thêm vào .  2. Pop () : Xoá 1 phần tử ở cuối mảng – trả về phân tử bị xoá .  3. Shift () : Xoá 1 phần tử ở đầu mảng – trả về phần tử bị xoá .  4. Unshift () : Thêm vào 1 phần tử ở đầu mảng – trả về độ dài  5. Join () : Chuyển mảng thành chuỗi ( cho phép sữa “join” (‘- ’) )  6. toString : Chuỗi mảng thành chuỗi  7. Reverse () : Đảo mảng  8. Sort () : Sắp xếp  9. Splice (st1, st2, st3) : thêm hoặc xoá phần tử ở vị trí bất kỳ trong mảng  10. Reduce () : Duyệt từng phần tử trong mảng , trả về 1 giá trị cuối cùng  11. Index of () : Tìm kiếm phần tử trong mảng |
| 34 | Phân biệt tham trị và tham chiếu trong JavaScript | **Tham trị :**  - Dùng cho kiểu DL “ nguyên thuỷ ”  - Sau khi gọi hàm giá trị ban đầu của đối số sẽ *không bị thay đổi*  **Tham chiếu :**  - Dùng cho kiểu DL “ đối tượng ”  - Sau khi gọi hàm giá trị ban đầu của đối số *có thể bị thay đổi*  (Đối với JS không có truyền tham chiếu) |
| 35 | Phân biệt giữa Hàm có return và hàm không có return | Return : được sd để trả về 1 giá trị nào đó cho lời gọi hàm  - Khi gặp từ khoá RETURN thì hàm trả về giá trị & lập tức thoát khỏi hàm  - 1 hàm có thể có nhiều từ khoá return nhưng phải nằm trong câu lệnh điều kiện . Tại mỗi thời điểm chỉ có 1 từ khoá return được thực thi  - Có thể return được nhiều kiểu DL  - Return nhiều giá trị cùng lúc thì dùng mảng |
| 36 | Lập trình hướng đối tượng là gì ? | OOP : Object Oriented Programming  - Là kỹ thuật giúp LTV có thể mô phỏng / ánh xạ đối tượng từ thực tế vào trong NNLT  - Đối tượng :  + Đặc điểm (Proprty): thuộc tính của đối tượng  + Hành vi (Behavior) : Chức năng của đối tượng |
| 37 | Phân biệt class và object | CLASS  - Là tập hợp các đối tượng có cùng thuộc tính với hành vi  - Là khuôn mẫu để tạo ra đối tượng  OBJECT  - Là thể hiện cụ thể của Class .  - Mỗi đối tượng sẽ có giá trị thuộc tính & hành vi khác nhau để phân biệt lẫn nhau . |
| 38 | Các đặc điểm trong Lập trình hướng đối tượng. Đưa ra 1 ví dụ 1 trong 4 tính chất. (lưu ý cần phải trình bày được 4 tính chất bằng tiếng anh) ? | **- Tính đóng gói (Encapsulation)** : là tính chất cho phép đảm bảo sự toàn vẹn DL của đối tượng ,sẽ ẩn giấu đi những chi tiết k được phép truy cập trực tiếp  **Mục đích :**  - Toàn vẹn DL , đảm bảo sự an toàn  - Để triển khai tính bao đóng , sử dụng : set / get  **VD:**  *Trong 1 vỉ thuốc có rất nhiều viên thước , nhưng mỗi viên thuốc lại được bao bọc trong ô nhỏ riêng và được bọc giấy bạc bên ngoài*  **- Tính kế thừa (Inheritance) :** cho phép lớp con (child / sub class) kế thừa lại các thuộc tính & hành vi đã được định nghĩa ở lớp cha (parent / super /base class) mà k cần phải khai báo lại .  - Lớp con có thể mở rộng thêm thuộc tính & hành vi mới  **Mục đích :** Tái sd mã nguồn , rút ngắn Source Code  **VD :**  *- Lớp Cha là smartphone, có các thuộc tính: màu sắc, bộ nhớ, hệ điều hành…*  *- Các lớp Con là iPhone, Samsung, Oppo cũng có các thuộc tính: màu sắc, bộ nhớ, hệ điều hành…*  **- Tính trừu tượng (Abstraction) :** là tính chất cho phép lược bỏ đi những thuộc tính , hành vi k cần thiết . Chỉ giữ lại những thành phần liên quan (hay chỉ quan tâm đến đầu vào & đầu ra , k quan tâm đến những xử lý chi tiết bên trong)  **Mục đích :** đơn giản hoá chương trình , tái sd mã nguồn  **VD:**  *- Một chiếc xe che giấu tất cả các cơ chế của nó nhưng cung cấp một cách để lái xe, tiếp nhiên liệu và duy trì nó cho chủ sở hữu của nó.*  *- Bất kỳ API nào ẩn tất cả các chi tiết triển khai của nó cung cấp dịch vụ cho các lập trình viên khác.*  **- Tính đa hình (Polymorphism)** : cùng 1 hành động / đối tượng nhưng có cách thể hiện khác nhau tuỳ vào ngữ cảnh/ thời điểm .  **-** Thể hiện thông qua Overload & Override (Module 2)  **VD :**  *Ở lớp smartphone, mỗi một dòng máy đều kế thừa các thành phần của lớp cha nhưng iPhone chạy trên hệ điều hành iOS, còn Samsung lại chạy trên hệ điều hành Android.* |
| 39 | Constructor là gì? Trong 1 class có nhiều hơn 1 constructor được hay không? | - Là hàm đặc biệt giúp khởi tạo đối tượng  - Mỗi Class chỉ có 1 contructor duy nhất (Chỉ trong JS)  => Gọi thông qua từ khoá “new” , chứa thuộc tính của đối tượng . Sd để tạo đối tượng  => Con function thì thông qua tên function (Function Name) . Sd để thể hiện hành vi , hành động của đối tượng  - Get (getter) : lấy thông tin hiện tại của đối tượng  - Set (setter) : thiết lập (thay đổi) giá trị hiện tại của thông tin đối tượng . |
| 40 | Những phương thức nào cho phép tương tác với chuỗi | - Tìm kiếm chuỗi con: indexOf(), lastIndexOf(), search()  - Cắt chuỗi con: slice(), substring(), substr()  - Chuyển hoa – thường: toUpperCase(), toLowerCase()  - Chuyển chuỗi sang mảng: split()  - Tìm kiếm kí tự trong chuỗi: charAt() |
| 41 | Các câu lệnh để vẽ 1 hình tròn. Giải thích từng câu lệnh? | - Khai báo HTML canvas để id width, height để khai báo vùng để vẽ  - Dùng script, lấy biến cavas và cho bằng document.getElementById  - Khai báo getContext 2D vẽ hình 2 chiều  - Để vẽ khai báo beginPath để bắt đầu  - Gọi hàm arc (tọa độ x, tọa độ y, bán kính,0 (góc bắt đầu), góc kết thúc (2PI (vẽ 1 hình tròn), PI(vẽ nửa đường tròn dưới), (PI – 0: nửa đường tròn trên))  - lineWidth: để chỉnh nét vẽ  - strokeStyle: để chỉnh màu  - stroke(): để vẽ |