|  |  |
| --- | --- |
| 1 | Các tính năng chính trong typescript?  1, Static Type ( là một thuộc tính của ngôn ngữ  lập trình, trong static typing, kiểm tra kiểu (type checking) được thực hiện trong thời gian biên dịch (compile time). Nó có nghĩa là kiểu dữ liệu (data type) của một biến (biến) đã được biết tại thời điểm biên dịch).  2, Interface (là sử dụng để kiểm tra, xem một đối tượng có phù hợp với một cấu trúc nhất định hay không,Bằng cách định nghĩa một interface, ta có thể    đặt tên trong trường hợp có sự kết hợp đặc biệt của các biến, đảm bảo rằng chúng luôn luôn đi cùng nhau).  3, Classes (là một khuôn mẫu để tạo các đối tượng. Chúng đóng gói dữ liệu bằng mã để làm việc trên dữ liệu đó. Các lớp trong JS được xây dựng trên nguyên mẫu nhưng cũng có một số cú pháp và ngữ nghĩa)   4, Modules(Nó phân chia code thành nhiều thành phần nhỏ và còn có khả năng tái sử dụng giúp dự án của bạn dễ tổ chức và dễ hiểu hơn so với file có hàng ngàn dòng code).  5, Generic (là cho phép cùng một hàm có thể chấp nhận các tham số với nhiều kiểu khác nhau. Việc tạo ra các thành phần có thể tái sử dụng với generics tốt hơn sử dụng kiểu any, vì generics bảo tồn kiểu của các biến vào và ra của chúng). |
| 2 | Lợi ích khi sử dụng typescript   * Hỗ trợ mạnh về OOP ( 4 tính chất : kế thừa, đóng gói, đa hình, trừu tượng, constructor, interface, class, ,.... * dễ dàng mở rộng dự án,  dễ bảo trì nâng cấp, … * được trợ nhiều framework * Dễ dàng phát hiện lỗi, hạn chế lỗi hệ thống |
| 3 | Trình bày các kiểu dữ liệu trong typescript?  -kiểu dữ liệu nguyên thủy ( string ,number, void,boolean, NaN, undefined)  -kiểu do người dùng tự định nghĩa( class, function, array, enum)` |
| 4 | Phân biệt continue, return, break   * continue: bỏ qua một bước, khi gặp lệnh này đoạn code sau đó sẽ bị bỏ qua mà chạy tới vòng lặp tiếp theo * break: thoát khỏi vòng lặp cho dù điều kiện vẫn đúng * return: thoát ra và trả về một giá trị |
| 5 | Trình bày về interface trong typescript  *Interface trong typescript cho phép bạn định nghĩ thuộc tính là gì và phương thức là gì mà đối tượng cần để được thực thi (implement). Nếu đối tượng tuân thủ đúng khuôn mẫu interface thì đối tượng đã implement interface ấy sẽ được thi hành đúng. Nếu interface không được thi hành đúng đắn thì typescript sẽ phát sinh lỗi ngay lập tức.*  so sánh lớp trừu tượng với interface  -giống nhau:   * đều không thể khởi tạo đối tượng * hàm abStract method thì không có phần thân   khác nhau:   * interface: -là một bản thiết kế của class, quy định các hành vi cho lớp triển khai, không được khởi tạo mà chỉ được tham chiếu bởi lớp thực thi. * -chỉ được gọi tên phương thức mà không được phép triển khai * -dùng từ khóa interface để khai báo và implement để triển khai phương thức và thuộc tính * -hỗ trợ đa kế thừa sử dụng từ khóa extends * - kế thừa bởi các lớp có ít liên quan đến nhau * - phạm vi truy cập là public   +abStract: là lớp cha định nghĩa phương thức cho các lớp con, dùng từ khóa abStract để khởi tạo,   chỉ hỗ trợ đơn kế thừa  kế thừa bởi các lớp có liên quan với nhau theo quan hệ cha con  phạm vi truy cập modifier |
| 6 | Làm thế nào để bạn gọi một phương thức khởi tạo của lớp cha từ một lớp con trong TypeScript?  -lớp con có từ khóa extends  -từ khóa super() gọi đến các thuộc tính, super. gọi đến phương thức |
| 7 | Làm thế nào để compile file typescript? |
| 8 | Các hàm Arrow / lambda trong TypeScript là gì?  -Arrow (lambda) cung cấp cú pháp ngắn gọn và thuận tiện để khai báo các hàm  Có 3 phần chính:   * tham số: một hàm có thể có các tham số tùy chọn * ký hiệu mũi tên ( => ) * câu lệnh: đại diện cho tập lệnh của hàm |
| 9 | Giải thích res parameter và quy tắc khai báo? Ví dụ   * là tính năng khai báo số lượng tham số không xác định khi khai báo hàm * sử dụng dấu ba chấm trước tham số đại diện ( …arr) * chỉ có thể là tham số cuối cùng trong danh sách |
|  |  |
| **Sử dụng được try-catch, xử lý được ngoại lệ** | |
| 10 | Ngoại lệ là gì? Tại sao phải xử lý ngoại lệ?   * là sự kiện không mong muốn xảy ra làm gián đoạn luồng chương trình bình thường * phải xử lý ngoại lệ  để đảm bảo chương trình không kết thúc đột ngột hoặc luồng thực thi chương trình không bị phá vỡ một cách không thân thiện |
| 11 | Xử dụng gì để bắt ngoại lệ?  sử dụng try-block sau đó là catch - block hoặc finally- block để bắt ngoại lệ:  try{  //các đoạn xử lý mà có thể phát sinh ngoại lệ  //phải có ít nhất 1 dòng code trong khối này  }  catch(error){  //xử lý lỗi ngoại lệ  //khối này có thể không cần thiết nếu finally được khai báo  }  finally{  //phần này có thể có hoặc không  khối này sẽ đc chạy sau khi xử lý xong try hoặc catch  } |
| 12 | Liệt kê các loại lỗi trong Typescript.  có 3 loại lỗi trong ts:  -lỗi cú pháp  - lỗi khi chạy chương trình  -lỗi logic |
|  |  |
| **Trình bày được mô hình Lập trình Hướng đối tượng** | |
| 13 | Nêu 4 đặc trưng của OOP? Lấy ví dụ  -tính kế thừa( inheritance): cho phép xây dựng lớp con cho phép kế thừa và tái sử dụng các thuộc tính và phương thức dựa trên lớp cha trước đó.  vd: lớp cha: smartphone bao gồm thuộc tính: màu sắc, bộ nhớ, hệ điều hành,..        lớp con : iphone, samsung,.. cũng có những thuộc tính này..  -tính đóng gói(encapsulation):cho phép che giấu thông tin và những tính chất xử lý bên trong của đối tượng. tăng tính bảo mật cho đối tượng và tránh tình trạng dữ liệu bị hư hỏng ngoài ý muốn  vd:  -tính đa hình( polymorphism ):cho phép những đối tượng khác nhau thực hiện những chức năng giống nhau theo những cách khác nhau.  vd: cùng câu lệnh “kêu đi” thì chó kêu : “ gâu gâu”  còn mèo kêu : “ meo meo”.  -tính trừu tượng(Abstraction ):loại bỏ các thuộc tính  và hành vi không liên quan.  vd: quản lý nhân viên thì chỉ cần: tên tuổi quê quán |
| 14 | Trình bày khái niệm OOP? Tại sao lại sử dụng OOP  - OOP (viết tắt của Object Oriented Programming)- lập trình hướng đối tượng là một ngôn ngữ lập trình dựa trên khái niệm về lớp và đối tượng.  tập trung vào các đối tượng thao tác hơn là logic để thao tác chúng  -tại sao lại sd:  +tối ưu việc quản lý source code;  +giúp tăng khả năng tái sử dụng và quan trọng hơn hết là giúp tóm gọn các thủ tục đã biết trước tính chất thông qua việc sử dụng các đối tượng. |
|  |  |
| **Sử dụng được lớp, đối tượng, phương thức, thuộc tính, hàm tạo** | |
| 15 | Hàm constructor được dùng để làm gì   * định nghĩa một hàm để khởi tạo đối tượng |
| 16 | Liệt kê access modifier?   * private: chỉ được truy cập trong 1 lớp * public: truy cập bất kỳ đâu * protected: truy cập theo kế thừa |
| 17 | Phân biệt static với this |
| 18 | Lớp là gì? Cách khai báo lớp  -lớp là trừu tượng hóa của đối tượng. những đối tượng có đặc tính tương tự nhau được sẽ tập hợp vào một lớp. |
| 19 | Triển khai được cơ chế nạp chồng phương thức (overloading) |
| 20 | Class và object giống và khác nhau thế nào? |
| 21 | Các thành phần của một lớp? |
| 22 | Vẽ sơ đồ lớp của lớp sau: tên lớp Customer: name, birthday; method: getInfo() : đưa ra tên + tuổi |
| 23 | Làm thế nào để khởi tạo đối tượng? |
| 24 | Có mấy cách khai báo lớp? |
| 25 | Khai báo lớp Customer vừa nêu |
| 26 | Làm thế nào để truy xuất thuộc tính của đối tượng? |
|  |  |
| **Triển khai được cơ chế kế thừa** | |
| 27 | Typescript có đa kế thừa không? |
| 28 | Thế nào là ghi đè? |
| 29 | Phân biệt abstract class và interface, khi nào sử dụng chúng? |
|  |  |
| **Sử dụng được UML (\*)** | |
| 30 | UML là gì? |
| 31 | Liệt kê các mối quan hệ giữa các đối tượng? Giải thích chúng? |
|  |  |
| **Tuân thủ Clean Code** | |
| 32 | Trình bày được tầm quan trọng của Clean Code |
| 33 | Kể tên một số Coding Convention trong Typescript |
|  |  |
| **Sử dụng được các kỹ thuật refactoring** | |
| 34 | Refactor là gì? Em đã sử dụng các kỹ thuật refactor nào? |
|  |  |
| **Sử dụng được các cấu trúc dữ liệu danh sách** | |
| 35 | Trình bày  khái niệm về data structures |
| 36 | Nêu cơ chế của cấu trúc Stack, Queue |