TYPESCRIPT

Interview Questions

By :- OMKAR TOMAR
Software developer

Ques 1:- Explain what TypeScript is and how it differs from JavaScript. Provide an example of a TypeScript feature that is not available in JavaScript.?

Ans :- TypeScript is a statically typed superset of JavaScript developed by Microsoft. It adds optional static type-checking to JavaScript, allowing developers to define types for variables, function parameters, and return values. The TypeScript code is then compiled to plain JavaScript, which can be executed in any JavaScript runtime.

One of the key differences between TypeScript and JavaScript is the type system. In TypeScript, you can explicitly specify the types of variables and function signatures, which enables catching type-related errors during development and provides better tooling support, such as code completion and type inference.

Ques 2:- Explain the concept of interfaces in TypeScript and provide an example of how they can be used.?

Ans :- Interfaces in TypeScript are used to define the structure of objects or classes. They describe the shape of an object, specifying the names of its properties and their corresponding types. Interfaces enable us to enforce a contract that objects must adhere to, ensuring better type checking and code consistency.

Example -:

```
interface Person {

firstName: string;

lastName: string;

age: number;

greet: () => void;

}

 class Employee implements Person {

 firstName: string;

lastName: string;

age: number;

constructor(firstName: string, lastName: string, age: number) {
```

```
this.firstName = firstName;
 this.lastName = lastName;
 this.age = age; }
 greet() {
 console.log(`Hello, my name is ${this.firstName} ${this.lastName}.`);
 }
}
const employee = new Employee("John", "Doe", 30);
employee.greet();
```

In this example, we define an interface `Person`, specifying the properties `firstName`, `lastName`, `age`, and a function `greet()`. Then, we create a class `Employee` that implements the `Person` interface, ensuring that it has all the required properties and methods defined in the interface.

Ques 3 :- Explain the concept of generics in TypeScript and provide an example of how they can be used. ?

Ans :- Generics in TypeScript allow us to create reusable components that can work with different types while preserving type safety. They provide a way to parameterize types, functions, and classes so that they can work with a variety of data types.

Example :-

```
function identity<T>(arg: T): T { return arg; }
 const numberValue: number = identity(5);
const stringValue: string = identity("Hello");
```

In this example, we define a generic function `identity<T>(arg: T): T`. The `T` within angle brackets represents a generic type. The function takes an argument of type `T` and returns the same type `T`. We can call this function with different types, such as `number` and `string`, and TypeScript infers the appropriate types for each call.

Ques 4 :- What is an interface in TypeScript?

Ans :- Interfaces define a contract or structure for objects that use that interface.An interface is defined with the keyword interface and it can include properties and method declarations using a function or an arrow function.

Example :-

```
Interface IEmployee {

empCode : number;

empName : String;

getSalary : (number) => number; // this is arrow function

getManagerName (number): string;

}
```

This is the example of interface how to create in typescript

Ques 4 :- What are modules in TypeScript?

Ans :- Modules in TypeScript are a collection of related variables, functions, classes, and interfaces.You can think of modules as containers that contain everything needed to execute a task. Modules can be imported to easily share code between projects.

```
Module Module_name {

Class xyz{

Export sum (x,y){

Return x+y;

}

}

}
```

Ques 5 :- What is the difference between the internal module and the external module?

Internal Module	External Module
Internal modules group the classes, interfaces, functions, variables into a single unit and can be exported in another module.	External modules are useful in hiding the internal statements of the module definitions and show only the methods and parameters associated with the declared variable.
Internal modules were a part of the earlier version of Typescript.	External modules are known as a module in the latest version.
These are local or exported members of other modules.	These are separately loaded bodies of code referenced using external module names.
Internal modules are declared using Module Declarations that specify their name and body.	An external module is written as a separate source file that contains at least one import or export declaration.

Ques 6 :- Does TypeScript support function overloading?

Ans :- Yes, TypeScript supports function overloading. But the implementation is odd. So, when you overload in TypeScript you only have one implementation with multiple signatures.

Example:-

```
class Customer {
name: string;
Id: number;
add(Id: number);
add(name:string);
add(value: any) {
if (value && typeof value == "number") {
//Do something
}
if (value && typeof value == "string") {
//Do Something
}
}
```

The first signature has one parameter of type number whereas the second signature has a parameter of type string. The third function contains the actual implementation and has a parameter of type any. The implementation then checks for the type of the supplied parameter and executes a different piece of code based on the supplier parameter type.

Ques 7 :- Explain Enum in TypeScript.?

Ans :- Enums or enumerations are a Typescript data type that allows us to define a set of named constants. Using Enums make it easier to document intent or create a set of distinct cases. It is a collection of related values that can be numeric or string values.

Example :-

```
Enum Gender {

Male,
Female
```

```
Other
}
console.log(Gender.Male); // Output: 0
//We can also access an enum value by it's number value.
console.log(Gender[1]); // Output: Female
```

Ques 8 :- What is an anonymous function?

Ans :- An anonymous function is a function that is declared without any named identifier. These functions are dynamically declared at runtime. Also, anonymous functions can accept inputs and return outputs, just as standard functions do. It is usually not accessible after its initial creation.

Example :-

```
let myAdd = function(x: number, y: number): number {
return a+b;
};
console.log(myAdd());
```

Ques 9 :- What is method overriding in TypeScript?

Ans :- If the subclass or the child class has the same method as declared in the parent class, it is known as method overriding. Basically, it redefines the base class methods in the derived class or child class.

Rules for Method Overriding:

- The method must have the same name as in the parent class
- It must have the same parameter as in the parent class.
- There must be an IS-A relationship or inheritance

Ques 10 :- What are getters/setters in TypeScript? How do you use them?

Ans :- Getters and setters are special types of methods that help you delegate different levels of access to private variables based on the needs of the program.

Getters allow you to reference a value but cannot edit it. Setters allow you to change the value of a variable but not see its current value. These are essential to achieve encapsulation.