**TypeScript**

1. **What is TypeScript, and how does it differ from JavaScript?**

TypeScript is a statically typed superset of JavaScript that compiles to plain JavaScript. It adds optional static typing, interfaces, enums, and other features to JavaScript.

1. **Explain the benefits of using TypeScript.**

TypeScript offers benefits such as type safety, enhanced tooling support, better code organization with interfaces and classes, improved maintainability, and early error detection during development.

1. **What are the basic types in TypeScript?**

Basic types in TypeScript include number, string, boolean, null, undefined, object, array, tuple, enum, and any.

1. **What is the difference between interface and type in TypeScript?**

Both interface and type can be used to define custom types. The main difference is that interface is more suitable for defining object shapes, while type can also define other types, such as unions, intersections, and aliases.

1. **Explain the concept of type inference in TypeScript.**

Type inference is TypeScript's ability to automatically deduce the type of a variable based on its initialization value. TypeScript analyzes the code and assigns types to variables, parameters, and return values without explicit type annotations.

1. **What is a union type in TypeScript? Provide an example.**

A union type allows a variable to hold values of multiple types. It is denoted using the | symbol between the types.

**Example:**

let value: string | number;

value = 'Hello'; // Valid

value = 10; // Valid

1. **What are generics in TypeScript, and why are they useful?**

Generics allow you to create reusable components and functions that work with multiple data types. They enable the creation of flexible and type-safe code by allowing types to be parameterized.

1. **Explain the never type in TypeScript.**

The never type represents the type of values that never occur. It is typically used to indicate functions that never return or variables that cannot have a value.

**Example:**

function throwError(message: string): never {

throw new Error(message);

}

1. **What are access modifiers in TypeScript, and how are they used?**

Access modifiers (public, private, protected) control the visibility and accessibility of class members in TypeScript. They determine how members can be accessed from outside the class.

1. **Explain the difference between interface and class in TypeScript.**

An interface is a blueprint for defining the structure of an object, while a class is a blueprint for creating objects with methods and properties. Interfaces define the shape of objects, while classes provide implementations.

1. **What is the difference between async/await and Promise in TypeScript?**

Both async/await and Promise are used for handling asynchronous operations. async/await provides a more synchronous way of writing asynchronous code, while Promise is based on chaining .then() methods to handle asynchronous operations.

1. **What is the readonly modifier in TypeScript, and how is it used?**

The readonly modifier in TypeScript indicates that a property can only be assigned a value once, either in the constructor or at the time of declaration.

**Example:**

class Person {

readonly name: string;

constructor(name: string) {

this.name = name;

}

}

1. **Explain the concept of namespaces in TypeScript.**

Namespaces are a way to organize code by grouping logically related classes, interfaces, functions, and variables under a single name. They help prevent naming conflicts and make code more modular and organized.

1. **What is the purpose of the import statement in TypeScript?**

The import statement is used to import functionality from other modules or files into the current module. It allows you to use classes, functions, variables, and other exports from external modules.

1. **Explain the difference between readonly and const in TypeScript.**

Both readonly and const are used to make variables immutable, but they have different use cases. readonly is used to make properties of objects immutable, while const is used to declare immutable variables.

1. **What is the as keyword in TypeScript, and how is it used?**

The as keyword is used for type assertions in TypeScript, where you explicitly specify the type of a value. It is often used when TypeScript cannot infer the correct type, or when working with union types.

**Example:**

let value: any = 'Hello';

let length = (value as string).length;

1. **Explain the concept of decorators in TypeScript.**

Decorators are a feature of TypeScript used to add metadata or behavior to classes, methods, properties, or parameters. They are prefixed with the @ symbol and are applied using special decorator functions.

1. **What is the purpose of the export keyword in TypeScript?**

The export keyword is used to export functions, classes, variables, or other declarations from a module, making them available for import in other modules.

1. **What are type guards in TypeScript, and how are they used?**

Type guards are TypeScript expressions used to narrow down the type of a variable within a specific block of code. They help improve type safety by performing runtime checks on the types of variables.

**Example:**

function isNumber(value: any): value is number {

return typeof value === 'number';

}

if (isNumber(myValue)) {

console.log('myValue is a number');

}

1. **Explain the concept of ambient declarations in TypeScript.**

Ambient declarations are used to define types for libraries or modules that are not written in TypeScript. They allow you to declare the shape of external JavaScript code and provide type information for TypeScript to use during compilation.

1. **What is the keyof operator in TypeScript, and how is it used?**

The keyof operator is a type operator in TypeScript used to obtain the union of keys of a given type. It is often used in conjunction with indexed access types to create type-safe object property accesses.

**Example:**

interface Person {

name: string;

age: number;

}

type PersonKey = keyof Person; // "name" | "age"

1. **Explain the concept of mapped types in TypeScript.**

Mapped types are a feature of TypeScript that allows you to create new types by transforming each property in an existing type in a specified way. They are useful for creating new types based on existing ones.

**Example:**

type Readonly<T> = {

readonly [P in keyof T]: T[P];

};

1. **What are namespaces and modules in TypeScript, and how do they differ?**

Namespaces are used for logical grouping of functionalities, while modules are used for code organization and isolation. Namespaces provide a way to structure code into smaller, more manageable pieces, while modules are used to encapsulate related code and dependencies.

1. **Explain the difference between const assertions and type assertions in TypeScript.**

const assertions are used to infer literal types for expressions, while type assertions are used to override the inferred type of a value. const assertions are more restrictive and ensure that the value is treated as a literal.

**Example:**

let value = 'Hello' as const; // const assertion

let length = (value as string).length; // type assertion

1. **What is the purpose of the never type in TypeScript?**

The never type represents the type of values that never occur. It is often used to indicate functions that never return or variables that cannot have a value. It is also useful in narrowing down types in conditional expressions.

**Example:**

function throwError(message: string): never {

throw new Error(message);

}

1. **Explain the concept of conditional types in TypeScript.**

Conditional types in TypeScript allow you to create types that depend on a condition. They enable you to define types that vary based on the types of other values.

**Example:**

type TypeName<T> = T extends string ? 'string' :

T extends number ? 'number' :

T extends boolean ? 'boolean' :

'unknown';

1. **What is the infer keyword in TypeScript, and how is it used?**

The infer keyword is used in conditional types to infer the type of a type parameter within the conditional type. It allows you to extract and use the inferred type within the conditional type.

Example:

type ExtractReturnType<T> = T extends (...args: any[]) => infer R ? R : never;

1. **Explain the concept of type guards and user-defined type guards in TypeScript.**

Type guards are TypeScript expressions used to narrow down the type of a variable within a specific block of code. User-defined type guards are custom functions that perform runtime checks on the types of variables and return a boolean value indicating whether the variable is of a certain type.

1. **What are the differences between interfaces and classes in TypeScript?**

Interfaces are used to define contracts for object shapes and do not provide implementations, while classes are used to create objects with methods and properties and provide implementations. Interfaces are purely a compile-time artifact and do not generate any JavaScript code, while classes are transpiled to JavaScript constructors and prototypes.

1. **Explain the concept of abstract classes in TypeScript and how they differ from interfaces.**

Abstract classes are classes that cannot be instantiated directly and may contain abstract methods that must be implemented by derived classes. Abstract classes can provide default implementations for methods, while interfaces cannot contain method implementations. Abstract classes can be used as a base for inheritance, while interfaces cannot be used for inheritance.