**General TypeScript Basics**

1. **What is TypeScript, and how does it differ from JavaScript?**  
   TypeScript is a superset of JavaScript that adds static typing and other features like interfaces, generics, and namespaces. Unlike JavaScript, TypeScript needs to be compiled into JavaScript.
2. **What are the advantages of using TypeScript?**
   * Static typing helps catch errors during development.
   * Better tooling support with IntelliSense.
   * Easier code refactoring.
   * Improved maintainability for large codebases.
3. **How do you install TypeScript in a project?**  
   Install TypeScript globally or locally using npm:

bash

Copy code

npm install -g typescript

1. **What is the purpose of the tsconfig.json file?**  
   It configures the TypeScript compiler options, like input/output files, strictness settings, and target JavaScript version.
2. **How can you compile a TypeScript file into JavaScript?**  
   Use the tsc command:

bash

Copy code

tsc filename.ts

**Type System**

1. **What are some basic types in TypeScript?**  
   number, string, boolean, array, tuple, enum, any, void, null, undefined, object, unknown, never.
2. **Explain the difference between any and unknown.**
   * any: Disables type checking for the variable, allowing any operation.
   * unknown: Restricts operations unless the type is explicitly checked or narrowed.
3. **What is type inference in TypeScript?**  
   TypeScript automatically infers the type of a variable based on its value:

ts

Copy code

let count = 5; // inferred as number

1. **What are type annotations? Provide an example.**  
   Type annotations explicitly specify the type of a variable:

ts

Copy code

let age: number = 25;

1. **What is the never type, and when would you use it?**  
   The never type represents values that never occur, such as a function that always throws an error or never returns:

ts

Copy code

function error(message: string): never {

throw new Error(message);

}

**Advanced Types**

1. **What are type aliases in TypeScript?**  
   Type aliases allow you to create custom type names:

ts

Copy code

type UserID = string | number;

1. **What is the difference between interface and type in TypeScript?**
   * interface: Used to define object shapes and supports extending.
   * type: More versatile, can alias any type, including unions and primitives.
2. **What are union and intersection types? Provide examples.**
   * Union: Combines multiple types, allowing one at a time:

ts

Copy code

let value: string | number;

* + Intersection: Combines types, requiring all at once:

ts

Copy code

type A = { x: number };

type B = { y: string };

type C = A & B;

1. **How does type assertion work?**  
   It overrides TypeScript's inferred type:

ts

Copy code

let value: any = "hello";

let strLength: number = (value as string).length;

1. **What is the difference between readonly and const?**
   * readonly: Used for properties in interfaces or classes to make them immutable.
   * const: Used for variables to prevent reassignment.

**Functions**

1. **How do you define the types of function parameters and return values?**

ts

Copy code

function add(a: number, b: number): number {

return a + b;

}

1. **What is the difference between optional and default parameters in TypeScript?**
   * Optional: Parameter may or may not be provided (?).

ts

Copy code

function greet(name?: string) {}

* + Default: Parameter has a default value if not provided:

ts

Copy code

function greet(name: string = "Guest") {}

1. **What is a function overload in TypeScript?**  
   Multiple function signatures for a single function implementation:

ts

Copy code

function add(a: number, b: number): number;

function add(a: string, b: string): string;

function add(a: any, b: any): any {

return a + b;

}

1. **How do you handle rest parameters in TypeScript?**  
   Use ... to accept multiple arguments:

ts

Copy code

function sum(...nums: number[]): number {

return nums.reduce((a, b) => a + b, 0);

}

1. **Explain the use of this in TypeScript functions.**  
   TypeScript can enforce the type of this in functions:

ts

Copy code

function show(this: { name: string }) {

console.log(this.name);

}

**Object-Oriented Features**

1. **How does TypeScript support object-oriented programming (OOP)?**  
   TypeScript supports classes, interfaces, inheritance, access modifiers, and abstract classes.
2. **What are access modifiers, and how are they used in TypeScript?**
   * public: Accessible anywhere (default).
   * private: Accessible only within the class.
   * protected: Accessible within the class and its subclasses.
3. **What is the difference between abstract classes and interfaces?**
   * abstract: Can have method implementations; used for base classes.
   * interface: Purely structural; defines only the shape of objects.
4. **Explain TypeScript’s readonly modifier in a class.**  
   Prevents modification of a property after initialization:

ts

Copy code

class User {

readonly id: number = 1;

}

1. **What are generics, and why are they useful?**  
   Generics provide type safety and reusability: