1. What is TypeScript?

- Superset of JavaScript → adds static typing.
- Compiles (transpiles) to JavaScript.
- Helps catch errors at compile time instead of runtime.

2. Static vs Dynamic Typing

- Static typing (TS): Variable types are fixed (let x: number = 5).
- Dynamic typing (JS): Variable types can change at runtime (let x = 5; x = 'hi';).

3. Type Annotations

- Explicitly declare variable types.
- Example: let age: number = 25; let name: string = 'Ali'; let isDone: boolean = true;

4. Functions with Type Annotations

- function add(x: number, y: number): number { return x + y; }
- Parameters and return types must be annotated.

5. Optional, Default, and Rest Params

- function greet(name: string, age?: number): void {} // optional
- function multiply(x: number, y: number = 2): number { return x * y; } // default
- function sum(...nums: number[]): number { return nums.reduce((a, b) => a + b); } // rest

6. Anonymous & Arrow Functions

- let square = function (x: number): number { return x * x; };
- let arrow = (x: number): number => x * x;

7. Type Alias

- type ID = string | number;
- let userId: ID = 123;

8. Literal Types

- let direction: 'up' | 'down';
- direction = 'up'; ■
- direction = 'left'; ■

9. Void & Never

- void: No return value → function logMessage(msg: string): void { console.log(msg); }
- never: Function never finishes (errors, infinite loops).
- Example: function throwError(): never { throw new Error('Error'); }

10. Enums

- enum Level { Kid = 1, Easy, Medium, Hard }
- let myLevel: Level = Level.Medium;

11. Union & Intersection

- Union → let id: string | number;
- Intersection → type Staff = Person & Employee;

12. Interfaces

- interface User { id: number; username: string; }
- let u: User = { id: 1, username: 'Ali' };

13. Classes in TS

- class Person { constructor(public name: string, private age: number) {} greet() { console.log(`Hello ${this.name}`); } }
- public → accessible anywhere
- private → only inside class
- protected → class + subclasses

14. Polymorphism

- class Animal { speak(): void { console.log('Some sound'); } }
- class Dog extends Animal { speak(): void { console.log('Bark'); } }

15. Generics

- function identity(value: T): T { return value; }
- let num = identity(5);
- let str = identity('hi');

Extra Notes

- Getters/Setters → control property access.
- Static methods/properties → belong to the class itself, not instances.
- Implements → class can enforce an interface.