Typescript Notes

TypeScript is a **typed superset of JavaScript**. It brings optional static typing, enabling **better tooling, code safety, and maintainability** — especially useful when scaling JavaScript codebases like MERN apps.

1️⃣ Basic Types

Basic types ensure that variables hold specific data types, catching type-related bugs during development.

Key Types:

- `string`, `number`, `boolean` - Primitive types
- `any` - Opts out of type checking (avoid when possible)
- `unknown` - Safer alternative to `any` (requires type checking)
- `never` - For functions that never return —like ones that always throw an error or run forever.
- `void` - Functions that return nothing
- `null` / `undefined` - Empty values
- `bigint` / `symbol` - Specialized types

```
// Explicit typing
let username: string = "Alice";
let age: number = 30;
let isDone: boolean = false;

// TypeScript can infer types without explicit annotations.
let isAdmin = false; // type inference

let anything: any = "disable type checking";

let userInput: unknown = getUserInput();
```

```
if (typeof userInput === "string") {
  console.log(userInput.toUpperCase());
}

let neverHappens: never;
function throwError(): never {
  throw new Error("Something went wrong!");
}

let nothing: void = undefined;

let bigNum: bigint = 12345678901234567890n;
let uniqueKey: symbol = Symbol("key");
```

2️⃣ Arrays and Tuples

Arrays: Collections of same-type elements

```
let scores: number[] = [95, 87, 92];
let names: Array<string> = ["Alice", "Bob"];

const nums: ReadonlyArray<number> = [1, 2, 3];
// nums[0] = 5; // Error
```

Tuples: Fixed-length arrays with specific types per position

```
let point2D: [number, number] = [10, 20];
let httpStatus: [number, string] = [200, "OK"];

// Named tuples (TypeScript 4.0+)
let user: [name: string, age: number] = ["Alice", 30];

// React's useState returns a tuple
const [state, setState]: [string, (val: string) => void] = useState("value");
```