**TypeScript ek superset hai JavaScript ka**, jo **Microsoft** ne develop kiya hai.  
Isme hum **optional types** use kar sakte hain, jaise ki **string, number, boolean** etc. TypeScript ka main advantage yeh hai ki yeh **compile-time pe errors** pakad leta hai, jabki JavaScript runtime pe errors show karta hai.  
Isse code zyada **clean, maintainable, aur less error-prone** hota hai. TypeScript ko hum **JavaScript ka supercharged version** samajh sakte hain.  
Yeh **React, Angular, Node.js** jaise frameworks ke saath bhi use hota hai.

**Key Features:**

1. **Static typing** - Type define kar sakte ho
2. **Interfaces and Classes** - Object-oriented programming ke concepts
3. **Compile-time error checking** - Development time pe error milta hai

Bhai, **JavaScript** aur **TypeScript** ke beech ka difference samjhate hain,

**1. Typing (Types ka use):**

* **JavaScript**: Dynamically typed language hoti hai, matlab variables ki type runtime pe determine hoti hai. Tu jo value dega, wo automatically type infer ho jayegi. Example:

let name = "Tinkal"; // Type ka pata nahi, runtime pe decide hoga

name = 23; // Yeh valid hai

* **TypeScript**: Statically typed language hai, matlab jab tu variable declare karta hai, toh uski **type specify karni padti hai**. Agar galat type assign karoge toh compile-time pe error milega. Example:

let name: string = "Tinkal"; // Type defined hai, sirf string assign kar sakte hain

name = 23; // Error milega, kyunki name ko string type diya hai

**2. Error Checking:**

* **JavaScript**: Errors **runtime** pe hi pata chalti hain. Matlab jab code run karega, tab hi pata chalega ki koi error hai.
* **TypeScript**: Errors **compile-time** pe hi pakad liye jaate hain. Matlab code likhne ke baad agar kuch galat hai, toh turant pata chal jaata hai.

**3. Type Safety:**

* **JavaScript**: Type safety nahi hoti, jo bhi value assign karna chahe, kar sakte ho. Toh agar galat type assign kar diya, code chalne pe issues aa sakte hain.
* **TypeScript**: Type safety hoti hai, matlab agar ek specific type define kar diya, toh wo type hi chahiye hoga. Agar kisi aur type ki value dene ki koshish karoge, error milta hai.

**4. Code Maintainability:**

* **JavaScript**: Agar code bahut bada ho, aur variables ki types define na ho, toh samajhna aur maintain karna mushkil ho sakta hai.
* **TypeScript**: TypeScript ka **static typing** aur **type inference** help karta hai code ko **zyaada maintainable** aur **error-free** banane mein. Badi projects ya teams ke liye ideal hai.

**5. Tooling Support:**

* **JavaScript**: Basic tooling support hota hai, jaise VS Code, jisme syntax highlighting aur code suggestions milte hain.
* **TypeScript**: TypeScript mein better tooling support milta hai, jisme **auto-completion**, **intellisense**, aur **error checking** rehta hai directly editor ke andar. Yeh developers ke liye bahut helpful hota hai.

**6. Compilation:**

* **JavaScript**: Directly browser ya Node.js pe run hota hai. **No need for compilation**.
* **TypeScript**: Yeh ek **compiled language** hai, matlab TypeScript ka code **JavaScript mein convert (transpile)** hota hai pehle, tab jaake wo run hota hai.

**7. Backward Compatibility:**

* **JavaScript**: Pure JavaScript code koi bhi browser ya environment mein directly run kar sakta hai.
* **TypeScript**: TypeScript ka code **JavaScript mein convert** hota hai pehle, toh **backward compatibility** hoti hai. Matlab TypeScript mein likha code purani JavaScript environments mein bhi run kar sakta hai.

**8. Frameworks & Libraries:**

* **JavaScript**: Mostly **React, Angular, Vue** mein JavaScript use hota hai.
* **TypeScript**: **React, Angular, Node.js** etc. sab major frameworks TypeScript ko support karte hain, aur TypeScript ka use **badi projects** mein zyada hota hai kyunki yeh better structure aur maintainability deta hai.
* An **array** is a data structure that stores multiple values in a single variable. These values are stored in a specific order and can be accessed using an index (starting from 0). Arrays can store values of the same type or different types.
* **Enum** ek aisa feature hai jisse hum **naam ke through value define kar sakte hain**, taaki hume number ya string yaad na rakhni pade.
* "Enum is used to assign names to numeric or string values, so the code is easier to read and understand."

**constructor**

* A **constructor** is a special method in a class that is automatically called when an object is created. It is used to initialize object properties.

**this** ka matlab hota hai **"ye object jisme ye function run ho raha hai."**

**this hamesha us object ko point karta hai jisse method call ho raha hai.**

Bhai, **interface** aur **type** dono hi TypeScript mein **custom types** define karne ke liye use hote hain — but unmein kuch subtle differences hote hain.

// Interface

interface Person {

name: string;

age: number;

}

// Type

type Person = {

name: string;

age: number;

}

**Rule of Thumb**

* Use interface when you are describing the **structure of an object**.
* Use type when you need **advanced type features** like union, intersection, or literal types.