**Summary 22 - 04 - 2025**

**Keyof in TypeScript**

keyof is used to store different types of values in the same variable and return them. It allows retrieving data from a type, such as Point, using a generic function.

type Point = { name: string; age: number };

type a = keyof Point;

let x:a;

x = "name";

x = "age";

interface Person{

   name: string;

   age: number;

   address: string;

   phone: string;

}

const person: Person = {

   name: "John Doe",

   age: 30,

   address: "123 Main St",

   phone: "123-456-7890"

};

**//generic function to get data from object**

function getData<T,K extends keyof T>(obj : T, key : K){

   return obj[key];

}

console.log(getData(person, "name"));

console.log(getData(person, "age"));

console.log(getData(person, "address"));

**Reduce Function**

The reduce() method iterates through an array, accumulating a total by applying a function to each element, where the first parameter (total) keeps a running sum and the second (num) represents the current value, starting with an initial value (in this case, 0).

**Method overloading**

This class in TypeScript demonstrates method overloading, where the speak() function is defined with two different signatures—one accepting a string and another accepting a number. However, since TypeScript only allows one implementation, the actual function uses the parameter type any to handle both cases dynamically. Inside the speak() method, it checks the type of arg using typeof().

**Example :**

class Meow {

    speak(s: string): string;

    speak(n: number): string;

    speak(arg: any): any {

        if(typeof(arg) === 'number') {

            return "meow number"

        }

if(typeof(arg) === 'string') {

            return "meow string";

        }

    }

}