***Day 5 Operators***

1. **Keyof operator**

The keyof operator in TypeScript is used to extract the keys of an object type as a union of string literal types. It provides a way to ensure type safety when working with object properties dynamically.

function getProperty<T, K extends keyof T>(obj: T, key: K): T[K] {

    return obj[key];

}

1. **Rest operator**

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

  return numbers.reduce((total, num) => total + num, 0);

}

const res = sum(1,2,3);

Collects all remaining arguments into an array. When you don't know how many arguments will be passed to a function, use rest operator.

**Rest Operator** is used in function **parameters**. **Spread Operator** (also ...) is used to **expand** elements, like:

sumof(...products)

1. **Overloading**

Define **multiple function signatures** for a single method, each with different parameter types or counts.

In TypeScript, **you define overloads with multiple function signatures**, and then provide **one actual implementation** that handles all cases.

speak(s: string): string;

speak(n: number): string;

speak(b: boolean): string;

These are the **overload declarations**. They tell TypeScript what calls are allowed

speak(arg: any): any {

    if (typeof arg === 'number') {

        return `Meow number ${arg}`;

    }

    if (typeof arg === 'string') {

        return `Meow string ${arg}`;

    }

    if (typeof arg === 'boolean') {

        return `Meow boolean ${arg}`;

    }

}

This is the **actual implementation** that handles **both overloads**. TypeScript only allows **one implementation**, and it must be compatible with all the declared signatures.

1. **Modules (import/export)**

Consider two files, one for importing and exporting. Now we will split into files and share using **import** and **export.**

//mathUtils.ts(a separate file)

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

    return a + b;

}

export const PI = 3.14;

export function area(radius: number): number {

    return PI \* radius \* radius;

}

//sub.ts(a separate file)

export default function subtract(a: number, b: number): number {

    return a - b;

}

When we use **export**, we are making **functions, variables, classes, or interfaces available to other files**. **default** is used when you're exporting **one main thing** from a module.

//app.ts(another file)

import {add, PI } from './mathUtils';

let result = add(10, 5);

console.log(`Result: ${result}`);

console.log(`Value of PI: ${PI}`);

//subfun.ts(a separate file)

import subtract from './sub.ts';

console.log(subtract(5,10));

When we use import to **bring in code that was exported from another module**.

If we want to **import everything** from a module use import \* as anyname from ‘./’;

import \* as MathUtils from './mathUtils';

console.log(MathUtils.add(5,10));

console.log(MathUtils.PI);

To Rename things when importing we can do it like this

import { multiply as mul } from "./1\_export";

console.log(mul(4, 5));