src\generics_intro.ts

```ts
//* Generics in TypeScript allow you to create reusable components or functions that can work with multiple data types.

function logAndReturn(value: number | string): number | string | boolean  {
    console.log(value);
    return value;
  }

const numberResult = logAndReturn(42);
const stringResult = logAndReturn("Hello, Generics!");
const booleanResult = logAndReturn(true);

console.log(numberResult);
console.log(stringResult);
console.log(booleanResult);




// Generic function to log and return an input value
function logAndReturn<T>(value:T):T {
    return value;
  }

// Using the generic function with different types
const numberResult = logAndReturn<number>(42);
const stringResult = logAndReturn<string>("Hello, Generics!");
const booleanResult = logAndReturn<boolean>(true);

console.log(numberResult);
console.log(stringResult);
console.log(booleanResult);

//! HomeWork Time
//! Here is the function overloading Example? Which I will tell you in tomorrow video
function add(a: number, b: number): number;
function add(a: string, b: string): string;
function add(a: any, b: any): any {
  return a + b;
}

const result1 = add(5, 10); // Output: 15
const result2 = add("Hello, ", "world!"); // Output: "Hello, world!"

//? You need to code the same using he Generics Types.



```