Generics are a feature of certain programming languages, including Java and C#, that allow you to write code that can work with multiple types of data, without specifying the exact data type until runtime. Generics provide a way to write type-agnostic code, meaning that the same code can be used with different types of data, such as integers, strings, or custom objects.  Generics are defined using angle brackets (<>) and a placeholder type parameter, which is replaced with a concrete type at runtime. For example, a generic list class could be written as List<T>, where T is the type parameter. When creating an instance of the list, you would specify the concrete type, such as List<Integer> or List<String>.  Generics allow you to write code that is more flexible and reusable, as it can be used with different types of data. They also provide a way to catch type-related errors at compile time, rather than runtime, improving the robustness and reliability of your code. Additionally, generics can make your code more readable and understandable, as the type information is contained in a single, central location, rather than scattered throughout the code.

Example :

function identity<T>(arg: T): T {

    return arg;

}

let output = identity<string>("hello");  // type of output is 'string'

console.log(output);

output = identity<number>(123);  // type of output is 'number'

console.log(output);

In this example, the identity function takes an argument of any type T, and returns a value of the same type T. The type parameter T is specified using angle brackets (<>) and is used to declare the type of the argument and the return value.  When you call the identity function, you can specify the type of T using a type argument, such as <string> or <number>. This allows the function to work with different types of data, such as strings or numbers, without having to create separate implementations for each type.  At runtime, the type argument is used to determine the concrete type of T, which is then used to type-check the argument and return value. This helps to catch type-related errors at compile time, improving the robustness and reliability of your code.