Typescript Generics

What is it:

Using generics in typescript expands the way you can write code as it offers a way to create components that you can re-use, and it can also work with several data types and isn't restricted to a single data type which makes programs more flexible.

Why use it:

Generic types have several advantages such as:

- Type-Safety: As it can hold a single type of object so no other objects will be stored by accident.
- No need to type cast
- Checked at compile time which reduces the runtime errors.

In TS you can use generic types in order to make a variable accept only the type that the user clarified so it counters the any datatype that is used in JS.

Possible Generics:

- Functions
- Methods
- Classes/Interfaces
- Variables

How to define it:

Generic types are strongly typed, you type a parameter between <> brackets using a special variable type <T> that denotes a type.
Ex:

```
function identity<Type>(arg: Type): Type { return arg;}
let myIdentity: <Type>(arg: Type) => Type = identity;
```

Part of the flexibility of generic types is that they have the ability to be used alongside non generic types.
Ex:

```
Function functionName<T>(param1:T ,param2:string):void{}
```

References:

- https://www.javatpoint.com/typescript-generics
- https://www.typescriptlang.org/docs/handbook/2/generics.html