GENERICS in TypeScript

Muhammad BIlal
/in/mrbilaltariq

WHAT ARE TYPESCRIPT GENERICS?

- Generics in TypeScript allow you to create reusable, flexible code that works with any data type. Instead of writing functions or classes that work with just one type, Generics allow you to specify types only when you use them.

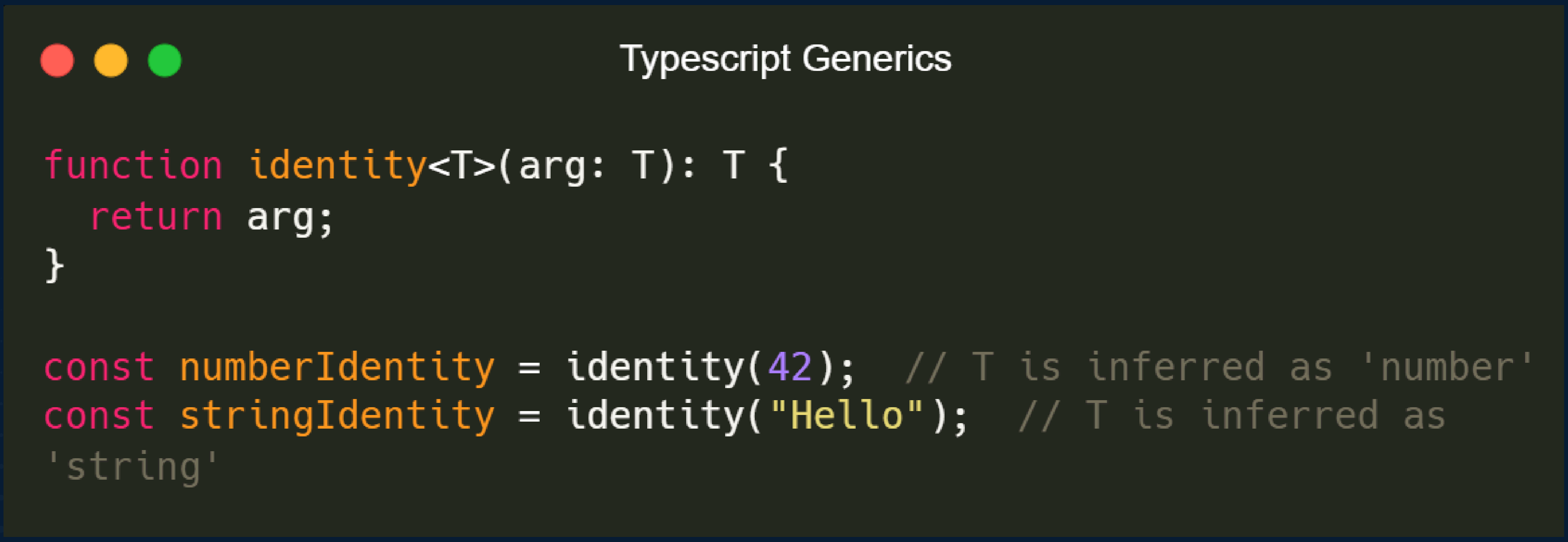
- In simple terms, Generics give you the ability to define placeholder types that can be used across different functions, classes, or interfaces.

HOW TO USE GENERICS

To use Generics, you define a placeholder type in the function, class, or interface using the angle brackets <>.

In this example:

- T is a placeholder type that allows identity() to handle any type.
- TypeScript automatically infers the type based on the argument passed in, but you can also specify it explicitly.



Typescript Generics

```typescript
function identity<T>(arg: T): T {
  return arg;
}

const numberIdentity = identity(42);  // T is inferred as 'number'
const stringIdentity = identity("Hello");  // T is inferred as
'string'
```

WHY SHOULD YOU USE GENERICS?

Here are some benefits of using Generics:

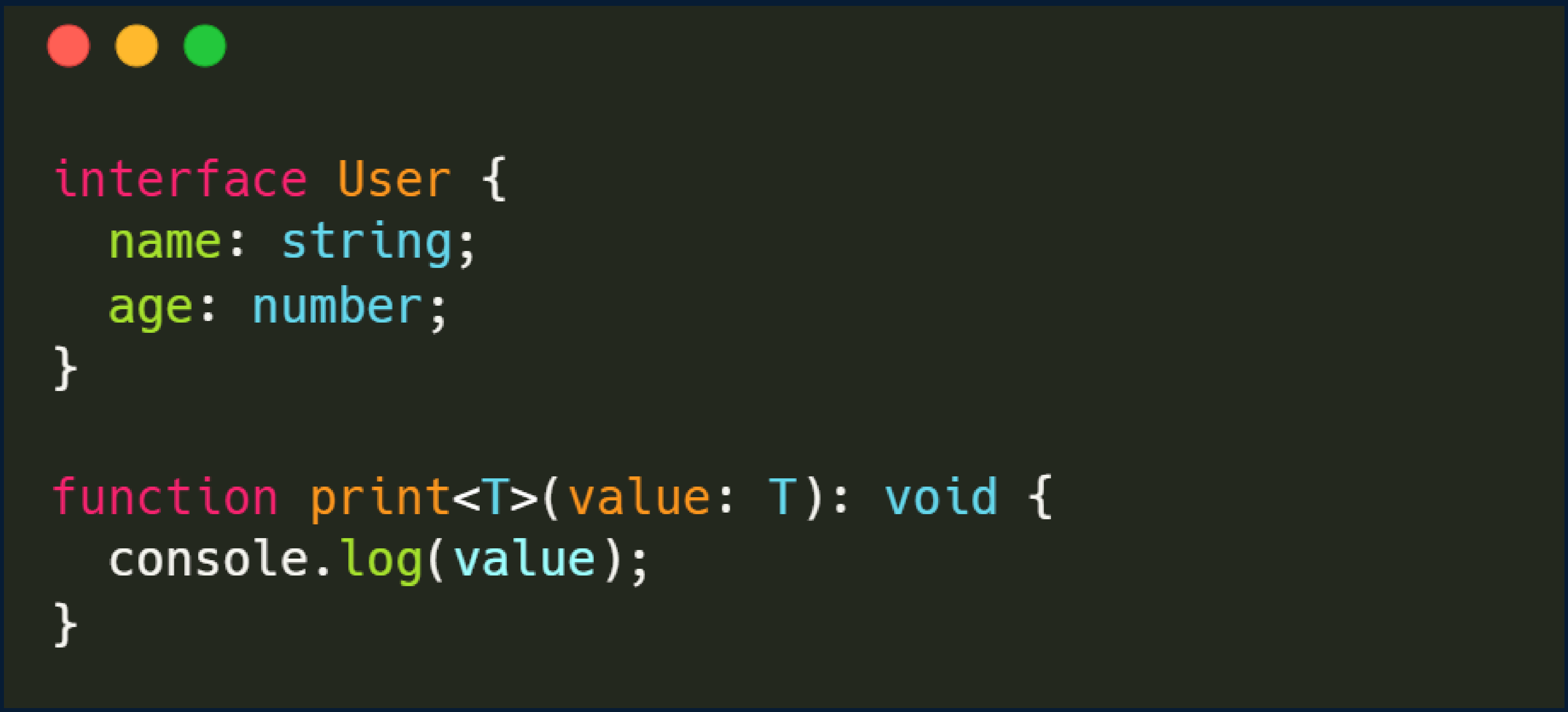
Flexibility: Generics allow functions and classes to work with any data type without sacrificing type safety.
Reusability: You can create functions or classes that can be used across different data types.
Type Safety: Even with flexible, reusable code, Generics maintain type checking and ensure your code is safe and bug-free.

GENERICS VS. INTERFACES

You might be wondering, can we achieve the same result with interfaces?

Generics and interfaces both define types, but they are used for different purposes.

Interfaces are great when you have a fixed structure and want to define the shape of an object.
Generics are more flexible and are used when the data type is not known in advance and can vary.



```
interface User {
  name: string;
  age: number;
}

function print<T>(value: T): void {
  console.log(value);
}
```

WHEN SHOULD YOU USE GENERICS OVER INTERFACES?

Use Generics when:

- You need flexibility and your code needs to work with multiple types (e.g., a function that works with numbers, strings, or objects).
- You want to create reusable, type-safe code without knowing the exact type upfront.

Use Interfaces when:

- You know the exact structure of the object you're working with and want to define a fixed blueprint.
- You need to define how specific types should behave (e.g., User, Product, etc.).

DO YOU USE GENERICS IN YOUR PROJECTS? WHAT’S YOUR EXPERIENCE WITH THEM?

Let me know your thoughts or any questions you have in the comments below!

Muhammad BIlal
/in/mrbilaltariq