Generics (Kiểu tổng quát)

Generics là cơ chế cho phép **tham số hóa kiểu dữ liệu**, tức là sử dụng **biến kiểu (type variable)** để trì hoãn việc xác định kiểu cụ thể cho đến thời điểm sử dụng. Cơ chế này giúp tái sử dụng cấu trúc kiểu trong nhiều ngữ cảnh khác nhau mà vẫn đảm bảo **an toàn kiểu (type safety)**.

Generics trong cấu trúc dữ liệu

Một ví dụ điển hình của generics là **Array**.
Nếu không sử dụng generics, một array có thể chứa bất kỳ kiểu giá trị nào, làm giảm khả năng kiểm tra kiểu.
Ngược lại, khi sử dụng generics, ta có thể mô tả chính xác **kiểu phần tử** mà array được phép chứa.

Ví dụ:

```
type StringArray = Array<string>;
type NumberArray = Array<number>;
type ObjectWithNameArray = Array<{ name: string }>;
```

Trong các định nghĩa trên:

- `StringArray` chỉ chứa các giá trị kiểu `string`
- `NumberArray` chỉ chứa các giá trị kiểu `number`
- `ObjectWithNameArray` chỉ chứa các object có property `name: string`

Nhờ đó, TypeScript có thể phát hiện lỗi ngay tại thời điểm biên dịch nếu một phần tử không phù hợp được đưa vào array.

Khai báo generics trong interface

TypeScript cho phép lập trình viên tự định nghĩa các kiểu dữ liệu sử dụng generics, thường gặp nhất trong **interface** và **class**.

Ví dụ:

```
interface Backpack<Type> {
  add: (obj: Type) => void;
  get: () => Type;
}
```

Ở đây, `Type` là một **biến kiểu**, đại diện cho kiểu dữ liệu sẽ được xác định khi interface `Backpack` được sử dụng. Interface này mô tả một cấu trúc tổng quát, không phụ thuộc vào kiểu cụ thể của dữ liệu mà nó lưu trữ.

Gán kiểu cụ thể cho generic

Khi sử dụng interface generic, ta cung cấp kiểu cụ thể cho biến kiểu đó:

```
declare const backpack: Backpack<string>;
```

Dòng lệnh `declare` cho TypeScript biết rằng tồn tại một hằng số tên `backpack` có kiểu `Backpack<string>`, mà không yêu cầu định nghĩa giá trị cụ thể của nó trong phạm vi hiện tại. Đây là một cơ chế thường dùng trong khai báo thư viện hoặc mô tả API bên ngoài.

Vì `Backpack` được khởi tạo với `string`, nên:

- Phương thức `get` sẽ trả về một giá trị kiểu `string`
- Phương thức `add` chỉ chấp nhận tham số kiểu `string`

Ví dụ:

```
const object = backpack.get();
```

Trong trường hợp này, `object` được TypeScript suy luận chính xác là `string`.

Kiểm tra kiểu với generics

Nếu cố gắng truyền một giá trị không phù hợp với kiểu đã khai báo cho generic, TypeScript sẽ phát hiện lỗi:

```
backpack.add(23);
```

Lỗi biên dịch:

```
Argument of type 'number' is not assignable to parameter of type 'string'.
```

Điều này cho thấy generics không làm suy yếu hệ thống kiểu; ngược lại, chúng cho phép biểu đạt các cấu trúc trừu tượng mà vẫn duy trì kiểm tra kiểu nghiêm ngặt.

Tổng kết

Generics cho phép:

- Trừu tượng hóa kiểu dữ liệu mà không đánh mất tính an toàn
- Tái sử dụng cấu trúc interface và class cho nhiều kiểu khác nhau
- Mô hình hóa chính xác mối quan hệ giữa dữ liệu đầu vào và đầu ra
- Giảm nhu cầu ép kiểu thủ công và tránh lỗi runtime

Trong TypeScript, generics là một trong những công cụ cốt lõi để xây dựng các hệ thống lớn, linh hoạt và có khả năng mở rộng cao, đặc biệt trong thiết kế thư viện, framework và API.