Structural Type System (Hệ thống kiểu cấu trúc)

Một trong những nguyên lý cốt lõi của TypeScript là **việc kiểm tra kiểu không dựa trên danh nghĩa (name) của kiểu**, mà dựa trên **hình dạng (shape)** của giá trị. Cách tiếp cận này thường được gọi là **duck typing** hoặc **structural typing**.

Trong **structural type system**, hai giá trị được xem là cùng kiểu nếu **chúng có cùng cấu trúc**, tức là cùng tập hợp các property cần thiết với kiểu dữ liệu tương ứng, bất kể chúng được khai báo hay tạo ra bằng cách nào.

Kiểm tra kiểu dựa trên shape

Xét ví dụ sau:

```
interface Point {
  x: number;
  y: number;
}

function logPoint(p: Point) {
  console.log(`${p.x}, ${p.y}`);
}
```

Interface `Point` mô tả một shape gồm hai property:

- `x: number`
- `y: number`

Khi gọi function:

```
const point = { x: 12, y: 26 };
logPoint(point);
```

Biến `point` **không hề được khai báo** là kiểu `Point`. Tuy nhiên, trong quá trình type-checking, TypeScript so sánh **shape của object literal** `point` với **shape của interface** `Point`. Do hai cấu trúc trùng khớp, đoạn mã được chấp nhận và thực thi hợp lệ.

Điều này thể hiện rõ đặc trưng của hệ thống kiểu cấu trúc: **tính tương thích kiểu được quyết định bởi cấu trúc, không phải bởi tên kiểu**.

So khớp một phần (subset matching)

Trong structural typing, việc so khớp shape **chỉ yêu cầu object có ít nhất các property cần thiết**, không yêu cầu phải giống hoàn toàn.

Ví dụ:

```
const point3 = { x: 12, y: 26, z: 89 };
logPoint(point3); // logs "12, 26"
```

Object `point3` có thêm property `z`, nhưng vẫn chứa đầy đủ `x` và `y`. Vì vậy, nó vẫn tương thích với `Point`.

Tương tự:

```
const rect = { x: 33, y: 3, width: 30, height: 80 };
logPoint(rect); // logs "33, 3"
```

Object `rect` có nhiều property bổ sung, nhưng vẫn thỏa mãn yêu cầu tối thiểu của `Point`.

Ngược lại, nếu object **thiếu các property bắt buộc**, TypeScript sẽ báo lỗi:

```
const color = { hex: "#187ABF" };
logPoint(color);
```

Lỗi xảy ra vì `color` không có `x` và `y`, nên không thỏa mãn shape của `Point`. TypeScript chỉ ra rõ rằng các property bắt buộc đang bị thiếu.

Class và object trong structural typing

Trong TypeScript, **không có sự khác biệt về mặt kiểm tra kiểu giữa class instance và object literal**. Cả hai đều được đánh giá dựa trên shape.

Ví dụ:

```
class VirtualPoint {
  x: number;
  y: number;

  constructor(x: number, y: number) {
    this.x = x;
    this.y = y;
  }
}

const newVPoint = new VirtualPoint(13, 56);
logPoint(newVPoint); // logs "13, 56"
```

Instance `newVPoint` của class `VirtualPoint` được xem là tương thích với `Point` vì nó có đầy đủ hai property `x` và `y` với kiểu phù hợp, mặc dù class không hề khai báo `implements Point`.

Kết luận

Trong **Structural Type System** của TypeScript:

- Kiểu dữ liệu được xác định bởi **cấu trúc**, không phải bởi tên hay nguồn gốc khai báo
- Object chỉ cần **chứa tập con các property bắt buộc** để được xem là tương thích
- Class và object literal được kiểm tra theo **cùng một nguyên tắc**
- Chi tiết cài đặt (implementation details) không ảnh hưởng đến việc so khớp kiểu

Cách tiếp cận này cho phép TypeScript:

- Chấp nhận JavaScript một cách tự nhiên
- Tăng tính linh hoạt trong thiết kế API
- Duy trì kiểm tra kiểu chặt chẽ mà không áp đặt mô hình danh nghĩa cứng nhắc

Đây là nền tảng quan trọng giúp TypeScript vừa giữ được tinh thần của JavaScript, vừa cung cấp một hệ thống kiểu mạnh mẽ và có khả năng mở rộng cao.