TS

▼ Các kiểu dữ liệu trong TS?

- **union**: cho phép sử dụng nhiều kiểu dữ liệu trong 1 biến
- **any**: khi không biết trước kiểu dữ liệu là gì
- **void**: dùng để thông báo function không có giá trị trả về
- **never**: giá trị đó sẽ không xảy ra. Được sử dụng khi chắc chắn việc gì đó không xảy ra
- **unknown:** giống như any nhưng ta không thể thực hiện bất kỳ thao tác nào khi mà chưa xác định type cụ thể của biến đó.
- **tuple**: có thể hiểu là kiểu dữ liệu mở rộng của array. Giúp chúng ta kiểm soát được thứ tự kiểu dữ liệu của các phần tử trong array.
- **intersection:** cho phép bạn kết hợp các thành viên của hai hoặc nhiều kiểu bằng cách sử dụng toán tử `&`. Điều này cho phép bạn kết hợp các kiểu hiện có để có được một kiểu duy nhất với tất cả các tính năng bạn cần.

▼ null và undefined trong TS?

- **null**: giá trị null cho biết không có giá trị. Một biến null không trỏ đến bất kỳ đối tượng nào. Do đó, bạn không thể truy cập bất kỳ thuộc tính nào trên biến hoặc gọi một phương thức trên đó.
- **undefined:** khi một biến được khai báo mà không tạo giá trị, nó sẽ được gán giá trị undefined.

▼ Enum trong TS?

- Enum là từ viết tắt của Enumeration (sự liệt kê), Enum dùng để định nghĩa kiểu dữ liệu với số lượng giá trị hữu hạn.

▼ Tự suy trong TS là gì ?

- TypeScript có thể tự suy kiểu của biến nếu bạn không cung cấp kiểu cụ thể.
- Diều này gọi là tự suy kiểu. Nó thường dùng khi các biến hoặc tham số được khởi tạo khi khai báo.

▼ **Interface và type khác nhau gì ?**

- Cả **Interface** và **type** trong ts cho phép bạn định nghĩ thuộc tính và phương thức là gì mà đối tượng cần để được triển khai (implement).
- Nếu đối tượng tuân thủ đúng theo khuôn mẫu thì sẽ được thực thi đúng ngược lại sẽ báo lỗi
- **Một số điểm khác nhau:**
  - **Type aliases có thể sử dụng computed properties**
    - Từ khóa `in` có thể được sử dụng để iterate tất cả các item bên trong một tập hợp keys. Chúng ta có thể sử dụng tính năng này để tạo mapped types.
  - **interface** có thể kế thừa từ 1 interface khác được còn **type** thì không
  - **Interface** có thể **merge**, **type** thì không. Nhiều khai báo có cùng tên chỉ hợp lệ khi sử dụng `interface`. Làm như vậy sẽ không ghi đè trước đó mà tạo ra kết quả hợp nhất chứa từ tất cả các khai báo

▼ **Generic trong TS?**

- Hiểu nôm na: kiểu dữ liệu mà có nhận tham số và trả về kiểu dữ liệu tương ứng.
- Hoặc hiểu đơn giản thì Generic type là việc cho phép truyền type vào components(function, class, interface) như là 1 tham số.Điều này sẽ giúp các components mềm dẻo hơn. Tái sử dụng tốt hơn.