Type Alias vs Interface 🔥

- **Type Alias**: "A type alias is exactly that - a name for any type" source
- **Interface**: "A way to name an object type" source

Primitive type

- Trường hợp dưới đây, chỉ đơn giản là mình tạo ra một cái tên gọi khác cho kiểu primitive.
- Cái này thì ít hữu dụng, vì cơ bản mình có thể dùng trực tiếp primitive type luôn cho khoẻ.

```
type StudentName = string;
type StudentAge = number;
type IsTopStudent = boolean;
```

Object type

- Add question mark **?.** for optional props.
- Add **readonly** to not allow updating value of a props (NOTE: still able to set on creation)

```
type Student = {
  readonly id: number;
  name: string;
  age?: number; // optional
}
```

```
interface Student {
  readonly id: number;
  name: string;
  age?: number; // optional
}
```

```
const studentA: Student = {
  id: 1,
  name: 'Alice',
}

const studentB: Student = {
  id: 2,
  name: 'Bob',
  age: 18,
}
```

Union type

- "A union type is a type formed from two or more other types, representing values that may be any one of those types" source
- Tạm dịch kết hợp 2 hoặc nhiều kiểu dữ liệu lại với nhau để tạo ra một kiểu dữ liệu mới.
- Cho phép giá trị có thể chấp nhận hoặc là kiểu dữ liệu này, hoặc là kiểu dữ liệu kia.

```
type Status = 'active' | 'inactive';
type ProductStatus = 0 | 1 | 2 | 3;
type StudentId = number | string;

interface Student {
  id: number | string;
  name: string;
  gender: 'male' | 'female';
  grade: 'A' | 'B' | 'C' | 'D' | 'E';
}
```

```
let x: number | string = 1;
x = 'super'; // works
x = 123; // works

x = false; // ts error: Type 'boolean' is not assignable to type 'string | number'.ts(2322)
```

Intersection type

```
interface BusinessPartner {
  name: string;
  credit: number;
}

interface Identity {
  id: number;
  name: string;
}

interface Contact {
  email: string;
  phone: string;
}

type Employee = Identity & Contact;
type Customer = BusinessPartner & Contact;
```

```
// interface way
interface Employee extends Identity, Contact {}
interface Customer extends BusinessPartner, Contact {}
```

```
type Customer = BusinessPartner & Contact;
let c: Customer = {
    name: 'ABC Inc.',
    credit: 1000000,
    email: 'sales@abcinc.com',
    phone: '(408)-897-5735'
};

type Employee = Identity & BusinessPartner & Contact;
let e: Employee = {
    id: 100,
    name: 'John Doe',
    email: 'john.doe@example.com',
    phone: '(408)-897-5684',
    credit: 1000
};
```

> Source: https://www.typescripttutorial.net/typescript-tutorial/typescript-intersection-types/

Add new props

- Able to add new props to **Interface**, but not **Type Alias**

```
interface Student {
  id: number;
  name: string;
}

// declaration merging
// works, no error
interface Student {
  age?: number;
}

const alice: Student = {
  id: 1,
  name: 'Alice',
}
```

```ts
type Student = {
  id: number;
  name: string;
}

// ts error: Duplicate identifier 'Student'.
type Student = {
  age?: number;
}
```

Cách đặt tên type vs interface

- Use **PascalCase** instead of **camelCase**. Use **Student** instead of **student**
- Don't use prefix **I**. Use **Student** instead of **IStudent**

```ts
interface Product {
  id: string;
  title: string;
}

interface ProductProps {
  data: Product;
}

function Product({ data }: ProductProps) {}
```

Nên dùng Interface hay Type

> For the most part, you can choose based on personal preference, and TypeScript will tell you if it needs something to be the other kind of declaration. If you would like a **heuristic**, **use interface until you need to use features from type**. source

- TL;DR Use Interface until You Need Type. source
- Always use interface for public API's definition source
- Preferring Interfaces Over Intersections. source
- Material UI types

Tham khảo

- https://medium.com/@martin_hotell/interface-vs-type-alias-in-typescript-2-7-2a8f1777af4c
- https://react-typescript-cheatsheet.netlify.app/docs/basic/getting-started/basic_type_example
- https://blog.logrocket.com/types-vs-interfaces-in-typescript/
- https://github.com/microsoft/TypeScript/wiki/Performance#preferring-interfaces-over-intersections

Series - Typescript cơ bản 🎉

- Tác giả: **Hậu Nguyễn**
- Được phát hành trên kênh youtube **Easy Frontend**.
- Tài liệu pdf và videos đều có bản quyền thuộc về Easy Frontend.
- Videos được phát hành cho fan cứng trước, public sau.
- Đăng ký fan cứng để xem series này đầy đủ và sớm nhất nhé.

☎️ Kết nối với mình

- ✅ Follow Facebook: https://www.facebook.com/nvhauesmn/
- ✅ Like Fanpage: https://www.facebook.com/learn.easyfrontend
- ✅ Youtube Channel: https://www.youtube.com/easyfrontend