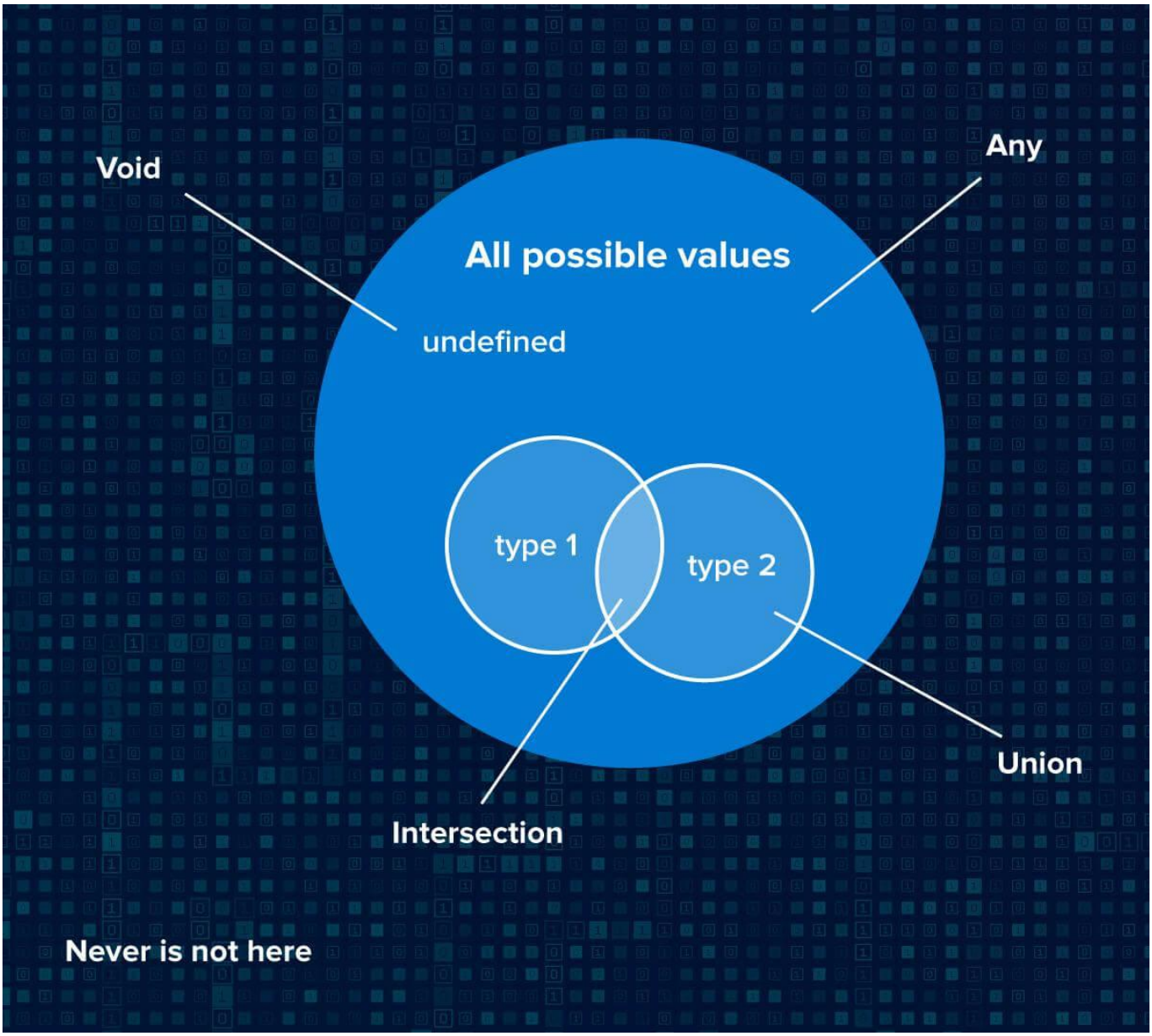
Type System Overview



> Source: https://serokell.io/blog/why-typescript

AGENDA

1. Type system overview
2. Type Alias vs Interface
3. Function
4. Enum
5. Generics
6. Utility types
7. And more

Các kiểu dữ liệu thường gặp

- Các kiểu dữ liệu bạn đã biết bên Javascript
  - **Primitive**: number, boolean, string, null, undefined, symbol
  - **Reference**: array, object, function
- Còn typescript, bạn sẽ bắt gặp: **any**, unknown, void, never, ...

```
let count = 5;
count = 'five';
// type error: Type 'string' is not assignable to type 'number'.ts(2322)
```

```
// adding any, solved 🤣
let count: any = 5;
count  = 'five'; // no error now :v
```

```
// three common primitive types: string, number and boolean
let count: number = 10;
let channelName: string = 'Easy Frontend';
let isActive: boolean = true;

// we can simply omit the type annotation
let count = 10;
let channelName = 'Easy Frontend';
let isActive = true;
```

Literal types

- Chỉ định một giá trị cụ thể làm kiểu dữ liệu.

```
let count: 1;
let channelName: 'easy';
let isActive: false;
let student: null;
```

```
let count: 1 = 2;
// error Type '2' is not assignable to type '1'
```

- Với **const**, khi omit type annotation, literal type sẽ được sử dụng. Vì const chỉ nhận được 1 giá trị, không thể thay đổi được.

```
const count = 1; // const count: 1
const channelName = 'Easy Frontend'; // const channelName: 'Easy Frontend'
const isActive = false; // const isActive: false
```

```
let count = 1; // let count: number
let channelName = 'Easy Frontend'; // let channelName: string
let isActive = false; // let isActive: boolean
```

```
const student = {
  id: 1,
  name: 'Easy Frontend',
}
// this is how ts understand:
// const student: {
//    id: number;
//    name: string;
// }
// because the props of an object can be updated

student.name = 'Typescript is easy! :P'; // works
```

```
const student = {
  id: 1,
  name: 'Easy Frontend',
} as const
// this is how ts understand:
// const student: {
//   readonly id: 1;
//   readonly name: "Easy Frontend";
// }
// you now can't update props of this object

student.name = 'Typescript is easy, really?!';
// error: Cannot assign to 'name' because it is a read-only property.
```

- Tương tự như object, array cũng có behavior tương tự.

```
const numberList = [1, 2, 3];
// ts understand: const numberList: number[]

const numberListAsConst = [1, 2, 3] as const
// ts understand: const numberListAsConst: readonly [1, 2, 3]
```

Type Assertion

```
function getStatusName(state: 'active') {
  console.log(state);
}

let s = 'active'; // let s: string
getStatusName(s)
// ts error: Argument of type 'string' is not assignable to parameter of
type '"active"'.ts(2345)

// solution 1
let s: 'active' = 'active'; // or const s = 'active'
getStatusName(s);


// solution 2
let s = 'active';
getStatusName(s as 'active')
```

> The reason why it's not called "type casting" is that casting generally implies some sort of runtime support. However, type assertions are purely a compile time construct and a way for you to provide hints to the compiler on how you want your code to be analyzed.

Tham khảo: https://www.typescriptlang.org/docs/handbook/2/everyday-types.html

---

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