Điều cần lưu ý khi làm việc với function

Default function return type

- Base on `return` keyword, typescript can inference the return type.

```
function sayHello() {
  console.log('Hi Easy Frontend');
}
// ts: function sayHello(): void
```

```
function sum(a: number, b: number) {
  return a + b;
}
// ts: function sum(a: number, b: number): number
```

Explicit return type

```
function sum(a: number, b: number): number {
  return a + b;
}
// ts: function sum(a: number, b: number): number
```

```
function sum(a: number, b: number): number {
  return a + b.toString();
}
// ts error: Type 'string' is not assignable to type 'number'
```

Optional and default parameter

- Add question mark **?** to tell typescript this is an optional parameter.
- Parameter cannot have question mark and initializer.

```
// optional parameter
// this is how ts understand
// function getLength(numberList?: number[] | undefined): number
function getLength(numberList?: number[]) {
  return Array.isArray(numberList) ? numberList.length : 0;
}

// default parameter
// this is how ts understand
// function getLength(numberList?: number[]): number
function getLength(numberList: number[] = []) {
  return Array.isArray(numberList) ? numberList.length : 0;
}

// ts error: Parameter cannot have question mark and initializer.ts(1015)
function getLength(numberList?: number[] = []) {
  return Array.isArray(numberList) ? numberList.length : 0;
}
```

Function Overload

> Tham khảo: https://www.typescriptlang.org/docs/handbook/2/functions.html#function-overloads

Other type: void and never

```
// function noop(): void
function noop() {
  return;
}

// function noop(): void
function noop() {
}
```

> The never type represents values which are never observed. In a return type, this means that the function throws an exception or terminates execution of the program.

```
type NewType = number & string; // never
```

```
function fail(msg: string): never {
  throw new Error(msg);
}
```

```
function fn(x: string | number) {
  if (typeof x === "string") {
    // do something
  } else if (typeof x === "number") {
    // do something else
  } else {
    x; // has type 'never'!
  }
}
```

> Tham khảo: https://www.typescriptlang.org/docs/handbook/2/functions.html#never

Destructuring parameter

```
function createStudent(id: number, name: string, age: number) {
  console.log(id, name, age)
}

createStudent(1, 'Bob', 18);
```

```
function createStudent(student: { id: number, name: string, age: number})
{
  const { id, name, age } = student;
  console.log(id, name, age)
}

createStudent({
  id: 1,
  name: 'Bob',
  age: 18,
});
```

```
function createStudent({ id, name, age }: { id: number, name: string, age:
number}) {
  console.log(id, name, age)
}

createStudent({
  id: 1,
  name: 'Bob',
  age: 18,
});
```

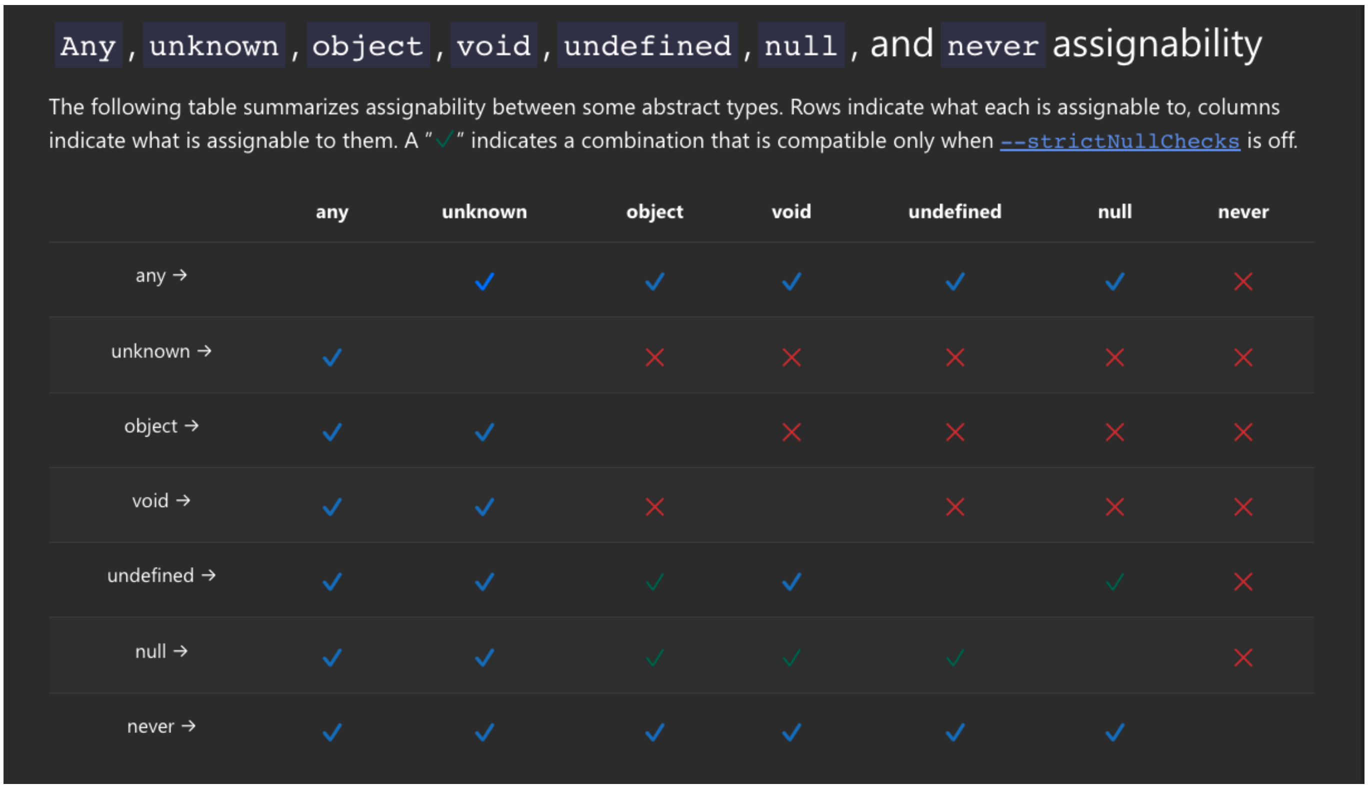
```
interface Student {
  id: number;
  name: string;
  age: number;
}

function createStudent({ id, name, age }: Student) {
  console.log(id, name, age)
}

createStudent({
  id: 1,
  name: 'Bob',
  age: 18,
} as Student);
```

Bảng type compatible

#	Type	Desc
1	string, number, boolean	3 kiểu primitive phổ biến
2	null	unavailable
3	undefined	not initialized
4	any	ignore type checking
5	unknown	not legal to do anything
6	void	function has no return
7	never	never return a value



`Any`, `unknown`, `object`, `void`, `undefined`, `null`, and `never` assignability

The following table summarizes assignability between some abstract types. Rows indicate what each is assignable to, columns indicate what is assignable to them. A "✔" indicates a combination that is compatible only when `--strictNullChecks` is off.

	any	unknown	object	void	undefined	null	never
any →		✓	✓	✓	✓	✓	✗
unknown →	✓		✗	✗	✗	✗	✗
object →	✓	✓		✗	✗	✗	✗
void →	✓	✓	✗		✗	✗	✗
undefined →	✓	✓	✔	✓		✔	✗
null →	✓	✓	✔	✔	✔		✗
never →	✓	✓	✓	✓	✓	✓	

> Source: https://www.typescriptlang.org/docs/handbook/type-compatibility.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