FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

LẬP TRÌNH TYPESCRIPT

TYPESCRIPT CƠ BẢN VÀ CÁC LOẠI DỮ LIỆU CƠ BẢN

THỰC HỌC – THỰC NGHIỆP

http://www.poly.edu.vn

Mục tiêu

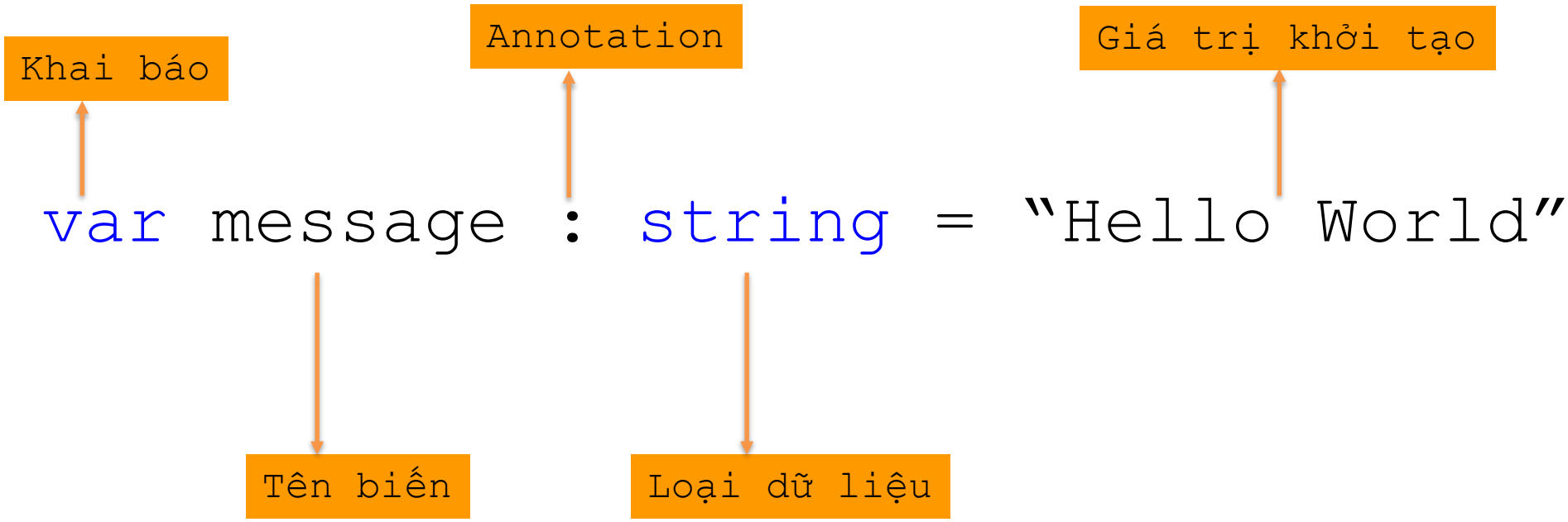
- Core types: number, string, Boolean
- Type inference
- Core types: object
- Core types: array
- Special types: tuple, any
- Type: union

Mục tiêu

- Literal type
- Null và undefined
- Unknown và any
- Type assertions

PHẦN 1

TYPE ANOTATION



CORE TYPE

number	1, 5.3, -10	Tất cả các số, không có sự khác biệt giữa Interger và float
string	`Hi`, "Hi", 'Hi'	Tất cả những giá trị kiểu text
boolean	true, false	Chỉ có 2 giá trị true hoặc false

```
let number1: number = 5;
let number2: number = 2.8;
let phrase: string = 'Result is ';
let permit: boolean = true;

const result = number1 + number2;
if(permit){
    console.log(phrase + result);
} else {
    console.log('Not show result');
}
```

TYPE INFERENCE

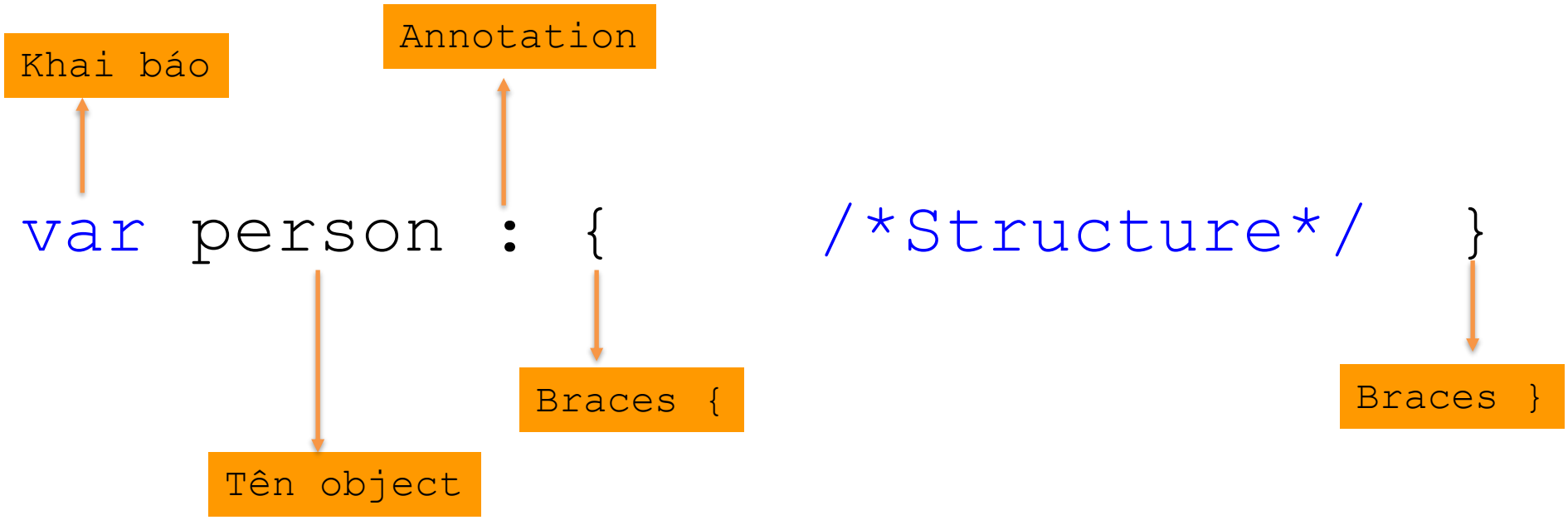
- Trong typescript, trình biên dịch typescript sẽ tự suy thông tin kiểu dữ liệu khi
  - Biến và các thành phần được cài đặt giá trị khởi tạo
  - Đặt giá trị mặc định cho tham số
  - Xác định giá trị trả về của hàm

```
function add(x=5)
{
    let phrase = 'Result is ';

    phrase = 10;
    x = '2.8';

    return phrase + x;
}
let result: number = add();
```

CORE TYPE

number	1, 5.3, -10	Tất cả các số, không có sự khác biệt giữa Interger và float
string	`Hi`, "Hi", 'Hi'	Tất cả những giá trị kiểu text
boolean	true, false	Chỉ có 2 giá trị true hoặc false
object	{age:30}	Bất kỳ object Javascript nào

TYPE ANNOTATION



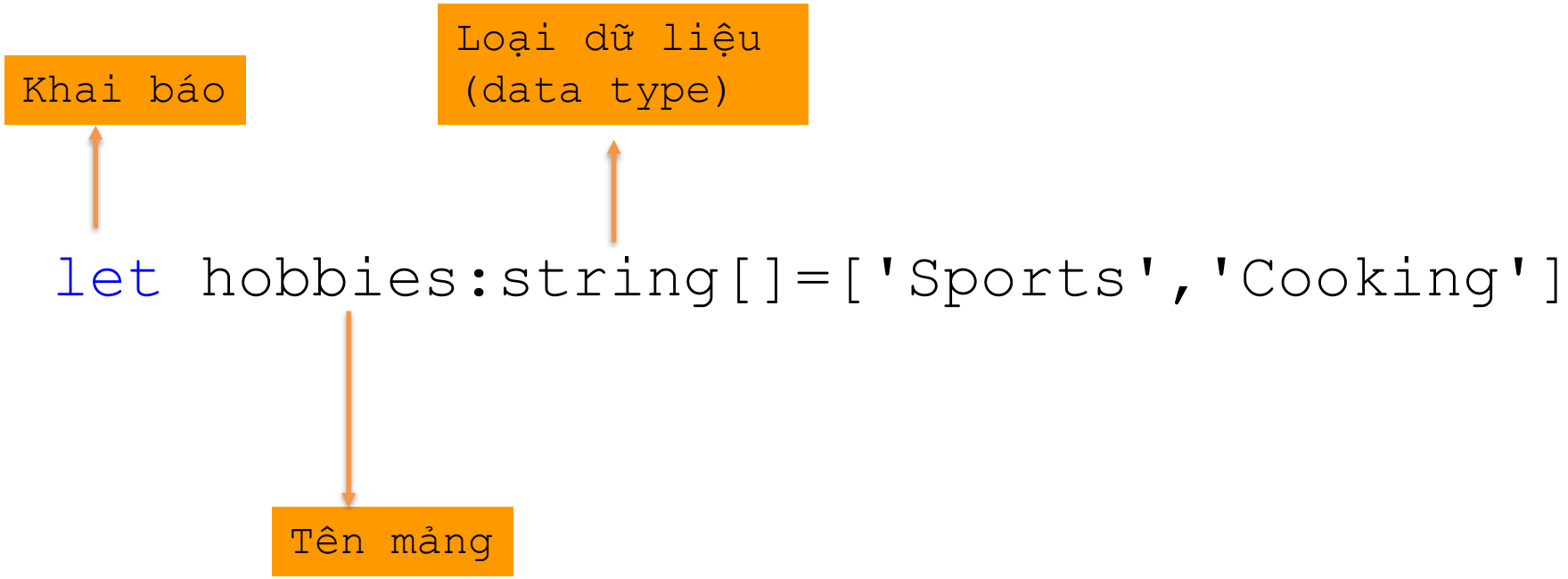
```
var person : {
    name: string,
    age: number
}

person = {
    name: 'Typescript',
    age: 11
}

console.log(person.name);
```


demo

CORE TYPE

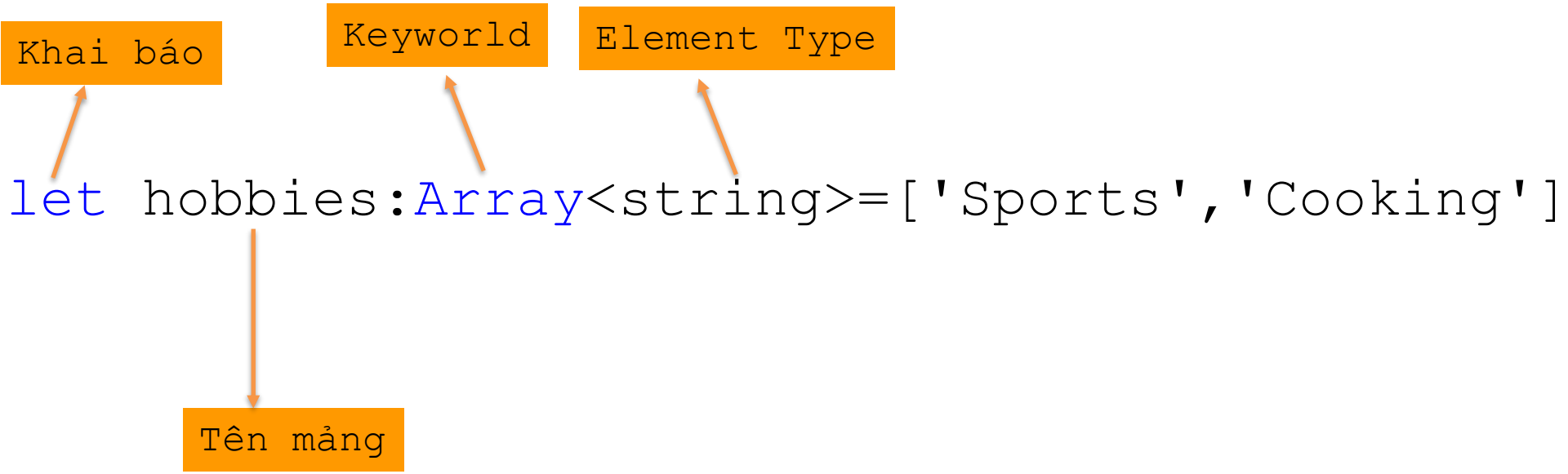
number	1, 5.3, -10	Tất cả các số, không có sự khác biệt giữa Interger và float
string	`Hi`, "Hi", 'Hi'	Tất cả những giá trị kiểu text
boolean	true, false	Chỉ có 2 giá trị true hoặc false
object	{age:30}	Bất kỳ object Javascript nào
array	[1,2,3]	Bất kỳ mảng Javascript nào (number, string,….)

Cách 1: Sử dụng cặp ngoặc vuông




```
let hobbies: string[]=['Sports','Cooking']
```

Cách 2: Sử dụng kiểu generic array




```
let hobbies: Array<string>=['Sports','Cooking']
```

SPECIAL TYPE

number	1, 5.3, -10	Tất cả các số, không có sự khác biệt giữa Interger và float
string	`Hi`, "Hi", 'Hi'	Tất cả những giá trị kiểu text
boolean	true, false	Chỉ có 2 giá trị true hoặc false
object	{age:30}	Bất kỳ object Javascript nào
array	[1,2,3]	Bất kỳ **mảng** Javascript nào (number, string,…)
tuple	[1,2]	**Mảng** các phần tử có kiểu dữ liệu khác nhau, (**Thêm mới bởi Typescript**), cố định độ dài của mảng

- Hàm hỗ trợ tuple
  - Push(): thêm phần tử
  - Pop(): xoá phần tử cuối cùng của tuple

```
let hobbies: [number, string]
hobbies = [2, 'Sports']
hobbies.push('Cooking');
```

SPECIAL TYPE

number	1, 5.3, -10	Tất cả các số, không có sự khác biệt giữa Interger và float
string	`Hi`, "Hi", 'Hi'	Tất cả những giá trị kiểu text
boolean	true, false	Chỉ có 2 giá trị true hoặc false
object	{age:30}	Bất kỳ object Javascript nào
array	[1,2,3]	Bất kỳ **mảng** Javascript nào (number, string,…)
tuple	[1,2]	**Mảng** các phần tử có kiểu dữ liệu khác nhau, (**Thêm mới bởi Typescript**), cố định độ dài của mảng
any	*	Thiết lập loại dữ liệu bất kỳ cho **biến / mảng** **(chưa biết kiểu dữ liệu)**

```
let hobby: any;
hobby = 2;
hobby = 'Cooking';
```

```
let hobbies: any[];
hobbies = [2, 'Sports', true]
```

PHẦN 2

TYPE

number	1, 5.3, -10	Tất cả các số, không có sự khác biệt giữa Interger và float
string	`Hi`, "Hi", 'Hi'	Tất cả những giá trị kiểu text
boolean	true, false	Chỉ có 2 giá trị true hoặc false
object	{age:30}	Bất kỳ object Javascript nào
array	[1,2,3]	Bất kỳ **mảng** Javascript nào (number, string,…)
tuple	[1,2]	**Mảng** các phần tử có kiểu dữ liệu khác nhau, (**Thêm mới bởi Typescript**), cố định độ dài của mảng
any	*	Thiết lập loại dữ liệu bất kỳ cho **biến / mảng** **(chưa biết kiểu dữ liệu)**
union	**Type1|Type2|.**	Thiết lập loại dữ liệu có thể cho **biến / mảng / hàm**

```
let hobby: string|number;
hobby = 2;
hobby = 'Cooking';
```

```
let hobbies: string[]|number[];
hobbies = ['Cooking', 'Sports'];
hobbies = [5,8,18,30]
```

TYPE

number	1, 5.3, -10	Tất cả các số, không có sự khác biệt giữa Interger và float
string	`Hi`, "Hi", 'Hi'	Tất cả những giá trị kiểu text
boolean	true, false	Chỉ có 2 giá trị true hoặc false
object	{age:30}	Bất kỳ object Javascript nào
array	[1,2,3]	Bất kỳ **mảng** Javascript nào (number, string,…)
tuple	[1,2]	**Mảng** các phần tử có kiểu dữ liệu khác nhau, (**Thêm mới bởi Typescript**), cố định độ dài của mảng
any	*	Thiết lập loại dữ liệu bất kỳ cho **biến / mảng** **(chưa biết kiểu dữ liệu)**
union	Type1|Type2|.	Thiết lập loại dữ liệu có thể cho **biến / mảng / hàm**
enum	Enum {NEW,OLD}	Tập hợp các constant (**Thêm mới bởi Typescript**)

```
enum Role {ADMIN, READ_ONLY,
AUTHOR };
```

```
{
  '0': 'ADMIN',
  '1': 'READ_ONLY',
  '2': 'AUTHOR',
  ADMIN: 0,
  READ_ONLY: 1,
  AUTHOR: 2
}
```

TS app.ts

TS app.ts > ...

```ts
enum Role {ADMIN, READ_ONLY, AUTHOR};
const person : {
    name: string,
    age: number,
    hobbies: string[],
    role: string,
    roletuple: [number, string]
} = {
    name: 'Typescript',
    age: 11,
    hobbies: ['Sports','Cooking'],
    role: Role.ADMIN, //Error
    roletuple: [2, 'author']
}

let favouriteActivites: any[];
favouriteActivites = [5, 'Sports', true];

if(person.role === Role.AUTHOR){
    console.log('is author');
}

person.roletuple.push('admin');
person.roletuple[1] = 10; //Error
person.roletuple = [0, 'admin', 'user']; //Error


```

- Literal type: giới hạn giá trị của một biến để hữu hạn các giá trị hợp lệ
  - Numeric literal types
  - String literal types
  - Boolean literal types
  - Enum literal types

TYPE

Keyworld

Custom name

```
type aliascustom = 'as-number'|'as-text'
```

type

```
let aliascustom = string | number
```

TS app.ts

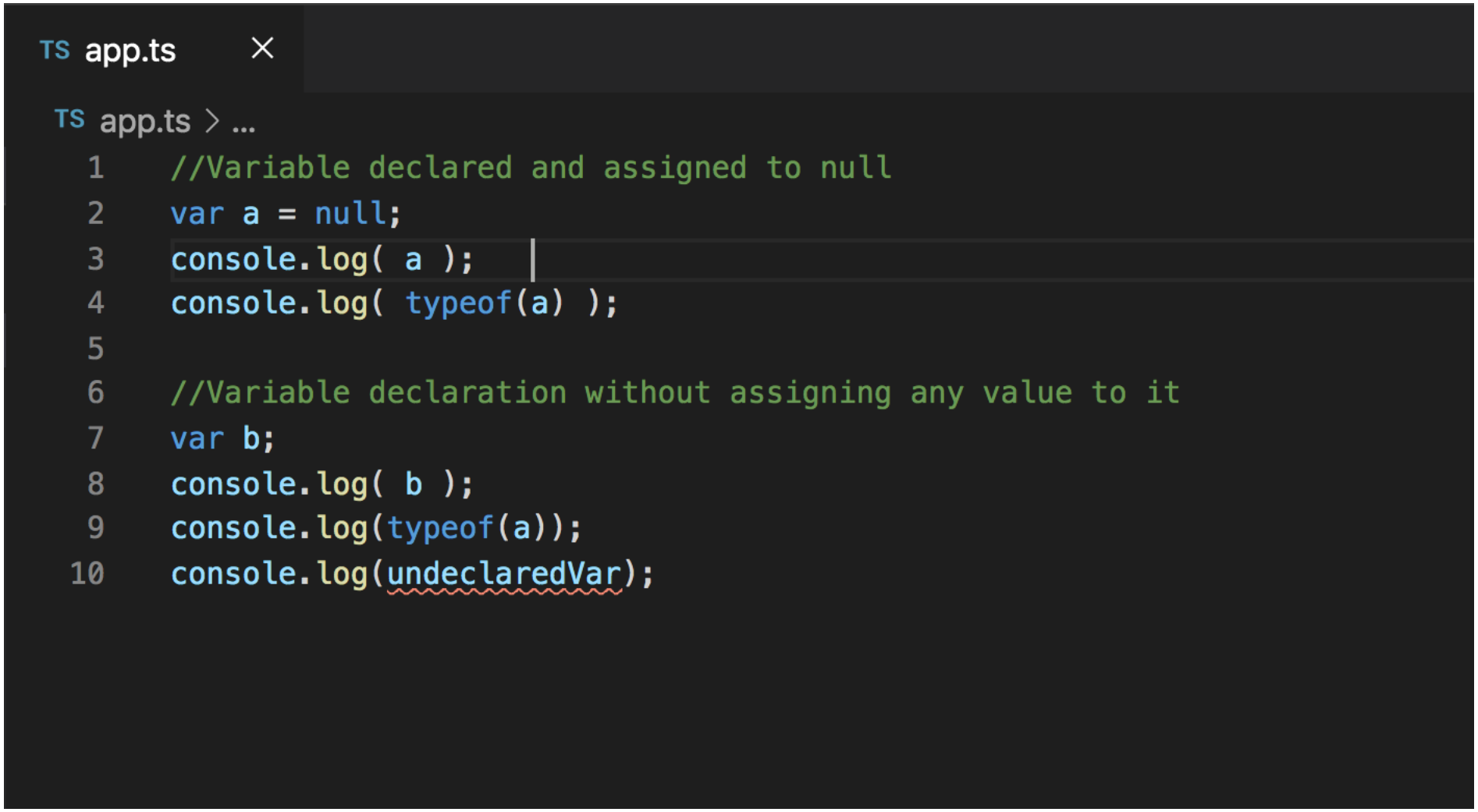
TS app.ts > ...

```ts
type Combinable = number | string;
function combine(input1: Combinable, input2: number|string, resultConversion: 'as-number'|'as-text'){
    let result;
    if(typeof input1==='number' && typeof input2==='number' || resultConversion==='as-number'){
        result = parseFloat(input1) + parseFloat(input2);
    } else {//concenated
        result = input1.toString() + input2.toString();
    }
    return result;
}

const combineNumber = combine(30, 26, 'as-number');
console.log(combineNumber);

const combineStringNumber = combine('30', '26', 'as-number');
console.log(combineStringNumber);

const combineText = combine('Typescript Vs ', 'Javascript', 'as-text');
console.log(combineText);
```

NULL & UNDEFINED

- Có giá trị gán cho biến
- Có thể gán cho biến không trỏ đến bất kỳ đối tượng nào
- Typeof là object
- Null có thể là rỗng hoặc không tồn tại
- Sự vắng mặt của giá trị cho một biến
- Null có thể được chuyển đổi thành 0

- Không có giá trị gán cho biến
- Biến được khai báo nhưng vẫn chưa gán giá trị
- Typeof là undefined
- Được dùng khi biến không được gán giá trị
- Sự vắng mặt của biến đó
- Có thể được chuyển đổi thành NaN


TS app.ts
TS app.ts > ...
1 //Variable declared and assigned to null
2 var a = null;
3 console.log(a);
4 console.log(typeof(a));
5
6 //Variable declaration without assigning any value to it
7 var b;
8 console.log(b);
9 console.log(typeof(a));
10 console.log(undeclaredVar);

UNKNOWN & ANY TYPE

- **Giống any**, biến kiểu unknown có thể được chỉ định bất kỳ giá trị nào
- any type cho phép thực hiện bất kỳ hoạt động nào mà không cần kiểm tra loại (check type)

TYPE ASSERTIONS (ÉP KIỂU)

```
let userInput: any = "this is a
string";
```

- Cách 1: sử dụng angle-bracket

```
let strLength: number = (<string>
userInput).length;
```

- Cách 2: sử dụng cú pháp as

```
let strLength: number = (userInput as
string).length;
```

TS app.ts

TS app.ts > ...

```ts
let userInput: unknown;
let userName: string;

userInput = 5;
userInput = 'Typescript';
userName = userInput;
userName = <string> userInput;
if(typeof userInput === 'string'){
    userName = userInput;
}
```


demo

SUMARRY

- ☑ Core types: number, string, Boolean
- ☑ Type inference
- ☑ Core types: object
- ☑ Core types: array
- ☑ Special types: tuple, any
- ☑ Type: union

SUMMARY

- ☑ Literal type
- ☑ Null và undefined
- ☑ Unknown và any
- ☑ Type assertions


thank
you!