WHY THIS CLASS ?

Does this code produce an error ?

```
let user = "name";

user = 4;

console.log(user);
```

Does this code produce an error ?


```
let user = "name";

user = 4;

console.log(user);
```

user should still be a string, not an integer !

It’s difficult to find errors with JavaScript

Can you identity easy the type of **dataPost**?

```
login: function () {
  if (this.username !== "") {
    this.loginFound = true;
  } else {
    this.loginFound = false;
  }
  localStorage.setItem("userName", this.username);
  let user = {
    user: this.username,
  };
  axios.post("http://localhost:5000/heartpost", user).then((response) => {
    this.dataPost = response.data;
    console.log(this.dataPost);
  });
},
logout: function () {
  this.loginFound = false;
  this.username = "";
  localStorage.setItem("userName", this.username);
```

Can you identity easy the type of **dataPost**?

NO !!

```
login: function () {
  if (this.username !== "") {
    this.loginFound = true;
  } else {
    this.loginFound = false;
  }
  localStorage.setItem("userName", this.username);
  let user = {
    user: this.username,
  };
  axios.post("http://localhost:5000/heartpost", user).then((response) => {
    this.dataPost = response.data;
    console.log(this.dataPost);
  });
},
logout: function () {
  this.loginFound = false;
  this.username = "";
  localStorage.setItem("userName", this.username);
```

1 It’s **hard** to **code fast** in JavaScript

Because JavaScript in an **untyped** language

untyped = no type declaration

In real life, we talk about **trees, table, pianos**

Not array of strings or dictionaries

2 Write code with real things

We would like to **define structures** to represent the world

A person is composed of a name, a age....

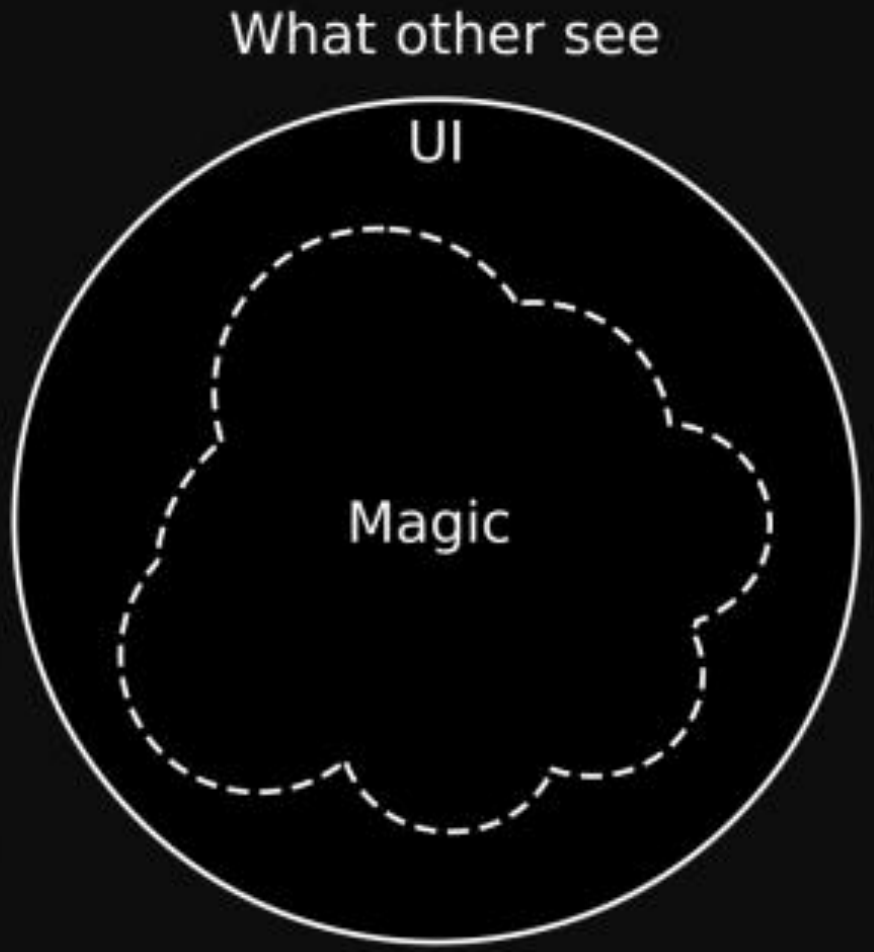
```
var app = new Vue({
  el: "#app",
  data: {
    URL: "http://localhost:3000",
    isNotLogin: true,
    formPost: false,
    loginUser: "",
    userName: "",
    pPost: "",
    filter: "allposts",
    postData: [],
    isPosting: true,
    indexToEdit: -1,
    searchFilter: "",
    resultOfsearchFilter: [],
    file: null,
    imgToUpload: "",
    urlImg: "",
    brainStormUrl: "",
    defaultProfile: "img/userProfile.j
    userProfile: "",
    users: [],
```

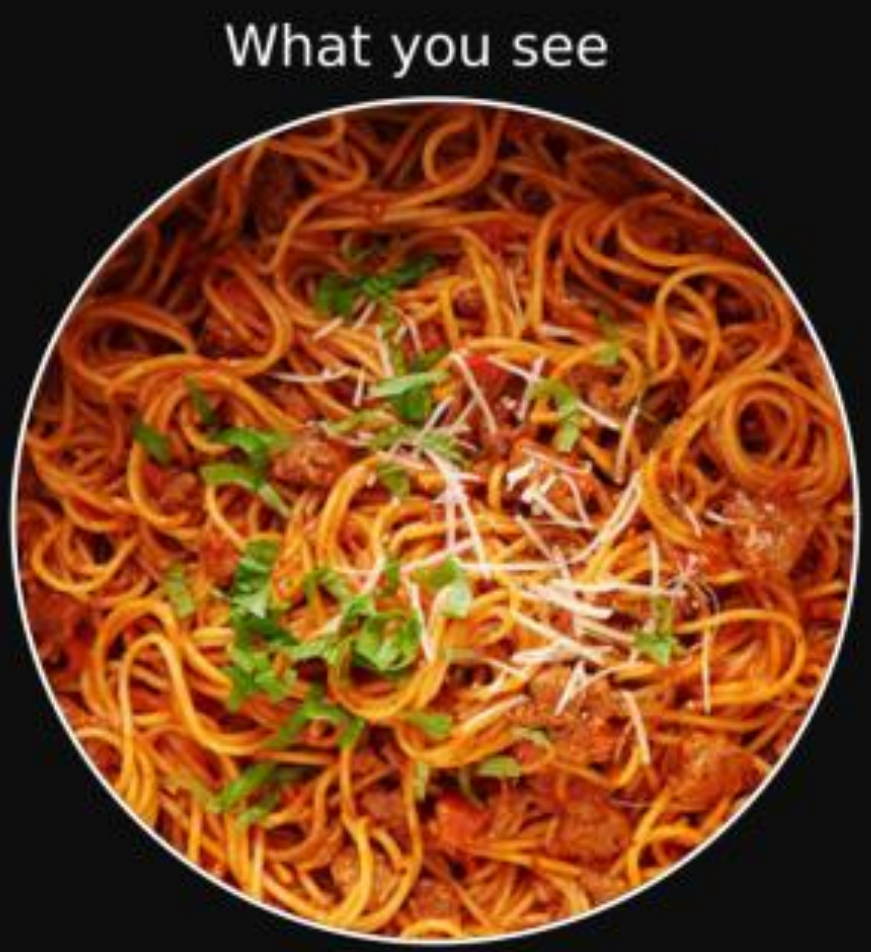
How do we manage

Large programs

with a lot of data to manipulate ?

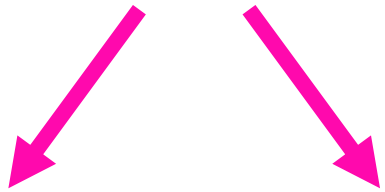
How to avoid **spaghetti code** ?





In Object-Oriented-Programming:

We group things in logical **units** (objects)



data **actions** on data

OOP

1. Typed languages
2. Object/Class
3. Aggregation
4. Encapsulation
5. Objects references
6. Inheritance
7. Polymorphism

Course evaluation

MAR

CHAPTER 1 - Typed languages

CHAPTER 2 - Object/Class

QUIZ 1 - 5 %

CHAPTER 3 - Aggregation

CHAPTER 4 - Encapsulation

QUIZ 2 - 5 %

CHAPTER 5 - Objects references

APR

CHAPTER 6 Inheritance

QUIZ 3 - 10 %

CHAPTER 6 Polymorphism

PROJECT

PROJECT - 30 %

MAY

EXAM

FINAL - 50 %

OOP

1. Typed languages
2. Object/Class
3. Aggregation
4. Encapsulation
5. Objects references
6. Inheritance
7. Polymorphism

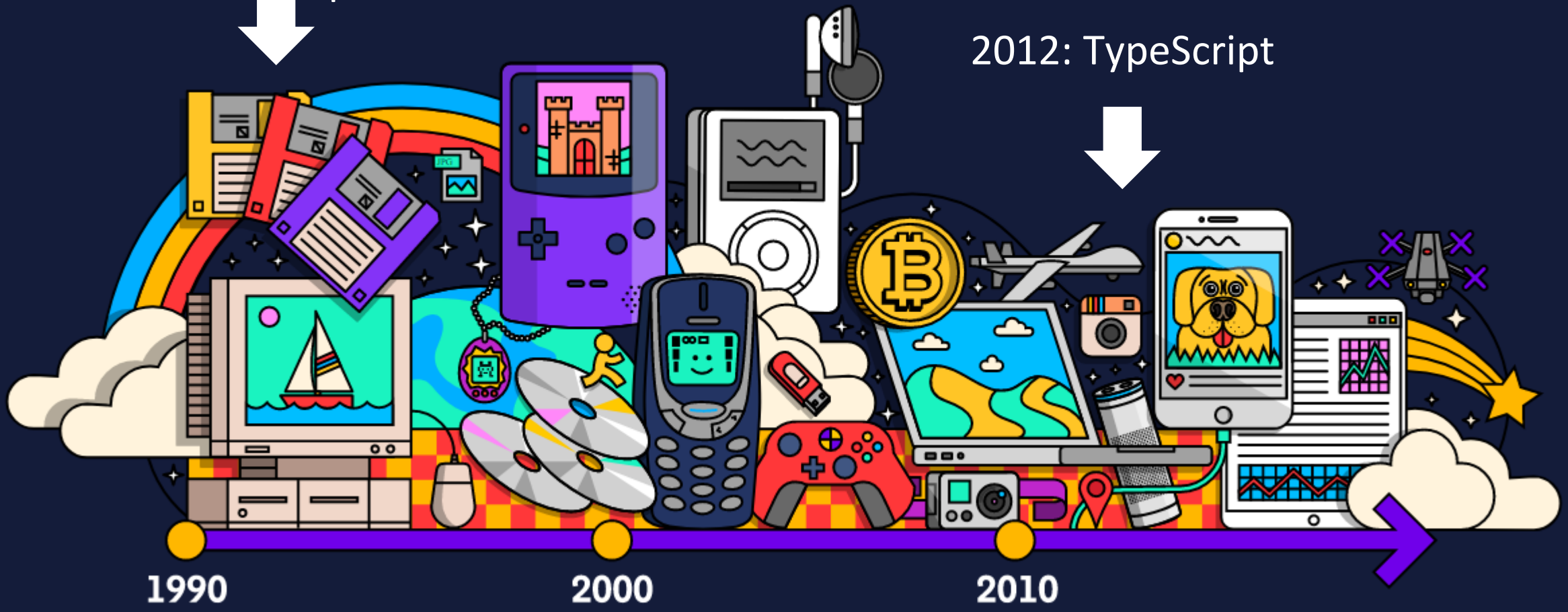
TYPED LANGUAGES

PREVENT ERRORS WITH TYPES

From **JavaScript** to **TypeScript**

1995 : JavaScript

2012: TypeScript



On VSCode, open a terminal and perform those 4 steps :

1 - Add NPM to the PATH

```
setx PATH "%PATH%;C:\Users\pros.nob\AppData\Roaming\npm;"
```

Your window account !!

2 – Install Typescript globally

```
npm install -g typescript
```

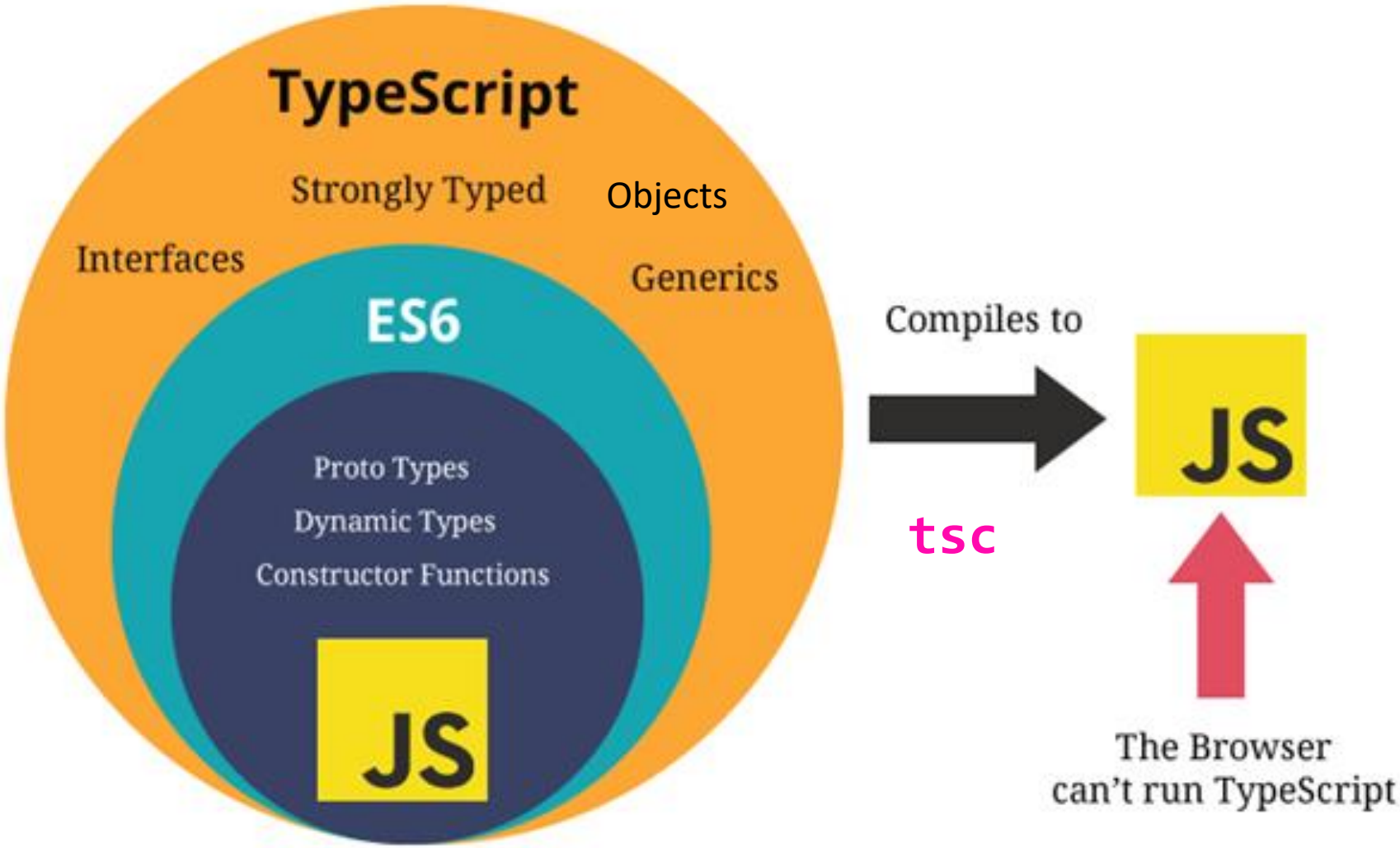
3 - Allow scripts to execute

```
Set-ExecutionPolicy -Scope CurrentUser  Unrestricted
```

4 – DONE !! Just check TypeScript works :

```
tsc --version
```

What is Typescript ?



What is Typescript ?

1- You can write **JavaScript** in Typescript !

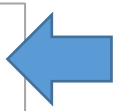
2- You need to **compile** your Typescript!

A compiler is a small program
To transform a language into
Another language

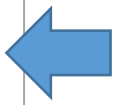
RULE 1- Typescript can **guess your type**

```
let user = "ronan";
user = 45;

console.log(user);
```



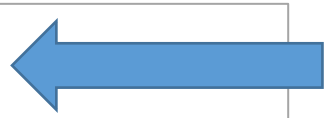
Typescript detects a type script



Typescript detects an error :
Cannot assign a integer to a string

RULE 2- If Typescript **cannot guess**, the type is : ***any***

```
let user;
```



Typescript does not detect anything, type is any

RULE 3 - Variables can have **annotations** after their names

ACTIVITY

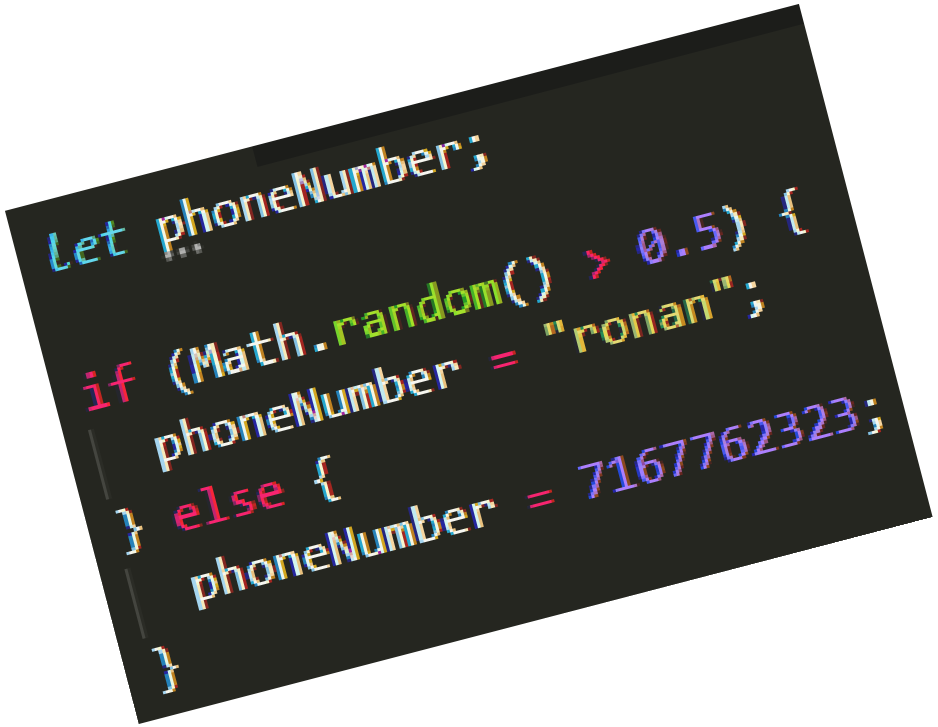
1- Open activity3.ts

2- Add an **annotation** to type phone Number

String or number !

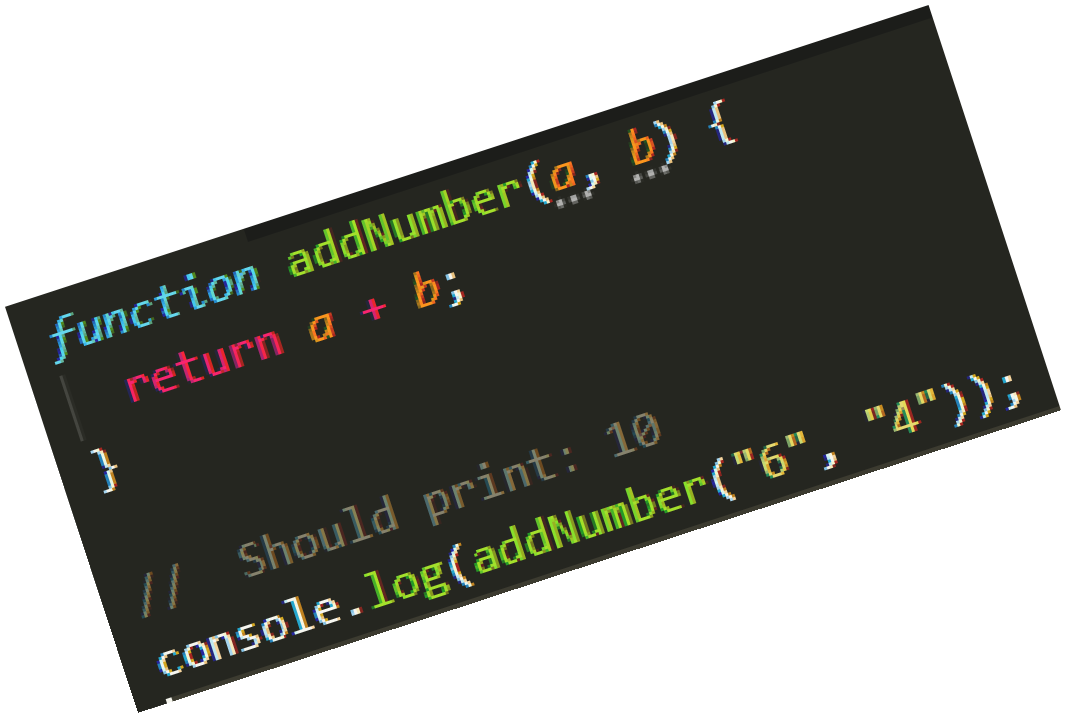
3- Check now, you have a mistake

And fix it !



ACTIVITY

1- Open activity4.ts



2- **Compile** : tsc activity4.ts

3- **Run** : node activity4.js

Check the result is not correct !

4- To fix this : add some type to the parameter a and b, to specify you want 2 numbers

5 – Fix the problem then with the “6” and “4”

ACTIVITY

1- Open activity5.ts

2- **Compile** : tsc activity5.ts

3- **Run** : node activity5.js

Check the result is not correct !

4- To fix this : add some type to parameters name and count in the function

5 – Then fix the problem in the function call : sayManyTimes(6, "Muriel");

RULE 4- You can choose if parameter are: **mandatory** or **optional**

MANDATORY

```
function printInfo (name: string, age: number) {
  console.log("name is " + name  + " age is " + age );
}

printInformation("ronan");
```

ERROR : because age is not provided

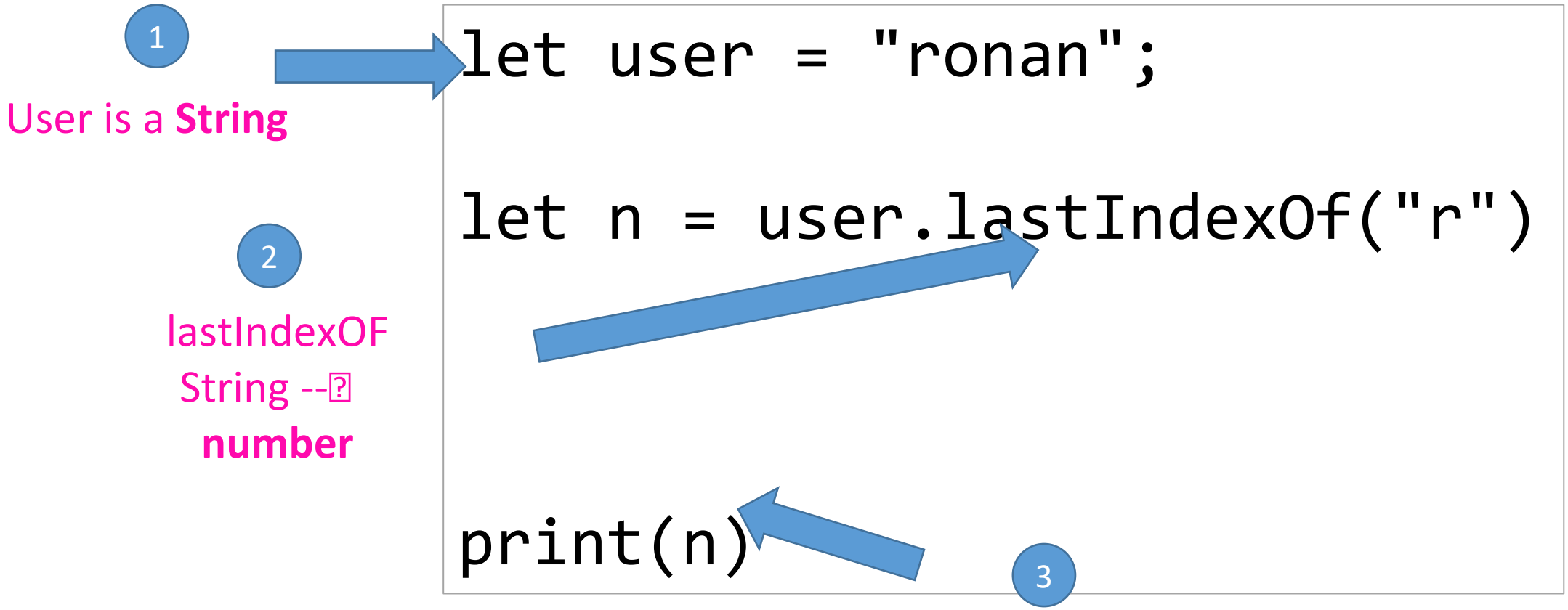
OPTIONAL

```
function printInfo (name: string, age?: number) {
  console.log("name is " + name  + " age is " + age );
}

printInformation("ronan");
```

NO ERROR because age is optional

RULE 5- Inference of types: TypeScript **can guess types**



Type code to make **the edition easier**

```
const myName = "ronan";

const size = myName.l
```

lastIndexOf
length (property) String.length: number
link
localeCompare
toLocaleLowerCase
toLocaleUpperCase

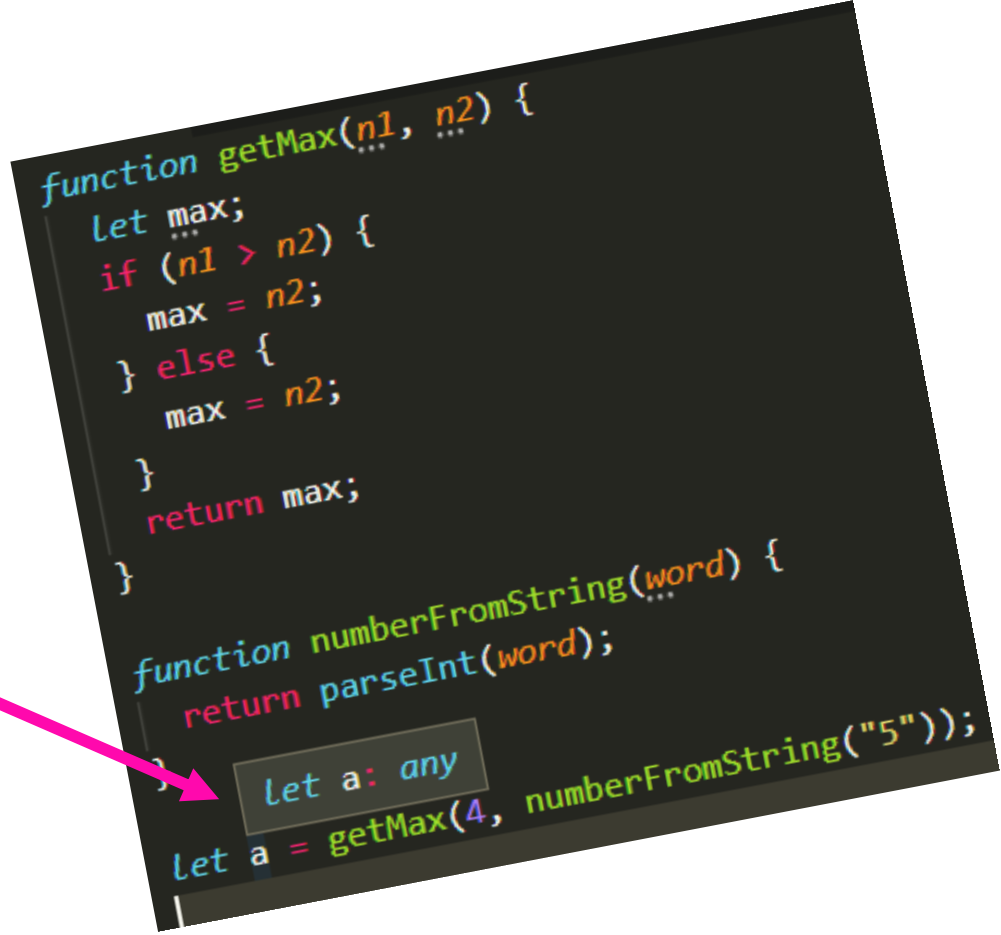
- ✓ myName is a **string**
- ✓ So while editing code, Typescript will display the **properties /functions related to string only**

ACTIVITY

1- Open activity6.ts

2- Right now the type of “a” is **any**

Because no types are specified in the code



3- Improve this : add types to parameters, function return, variables ….

4– Then check that the display type *(when mouse hover)* of a is : **number**

TYPE SYNTAX ,

1- PRIMITIVE TYPES !

```
let name : string = "ronan";
```

A string

2 ARRAY TYPES !

```
let nums : number[] = [5, 8];
```

An array of numbers

3- UNION OF TYPES !

```
let name : string |boolean = "ronan";
```

Can be a string or a boolean

```
let nums : (number|string)[] = [5,"8"];
```

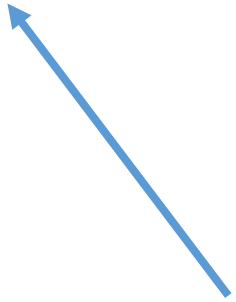
An array of number or strings

4- OBJECT TYPES !

```
let student : { x: number; y: number } = {x : 45, y : 55};
```

An object composed of 2 numbers

```
let students : { x: number; y: number }[] = [
                                              {x : 45, y : 55},
                                              {x : 99, y : 77}
                                            ]
```



An array of object composed of 2 numbers

5- ALIAS TYPES !

We want to use the same type more than once and ***refer to it by a single name.***

BEFORE

```
let sun: { name: string;  age : number }
=   { name: 'vun', age: 17 }

let som: { name: string;  age : number }
=   { name: 'som', age: 48 }
```

AFTER

```
type Student = { name: string;  age : number }

let sun:Student= { name: 'vun', age: 17 }

let som:Student= { name: 'som', age: 48 }
```

Example :

create an alias to define a Point (x, y)

```
type Point = {
  x: number;
  y: number;
};

// Exactly the same as the earlier example
function printCoord(pt: Point) {
  console.log("The coordinate's x value is " + pt.x);
  console.log("The coordinate's y value is " + pt.y);
}

printCoord({ x: 100, y: 100 });
```

Does this code produce an error ?

```
let student : { age: number; address: string} =
                { age : 19,
                  address : 45
                };
```

A - YES B - NO

Does this code produce an error ?

```
let nums : (number|boolean) [] = [5, false, 58];
```

A - YES B - NO

Does this code produce an error ?

```
let nums : (number|boolean) [] = [5, false, "58"];
```

A - YES B - NO

Does this code produce an error ?

```
let students : { age: number; address: string} [] =
               [  {    age : 19, address : “paris “},
                  {    age : 15, address : “chicago “},
               ];
```

A - YES B - NO

Does this code produce an error ?

```
Type Result= {
          topic: string;
          score: number;
          student: string;
}

Let ronanResult : Result = {topic: ‘java’, score:45 }
```

A - YES B - NO

TO SUM UP

On this course we will improve your code style :

- **Type data** to avoid mistakes
- **Group data into objects** to avoid spaghetti code

RULE 1 - Typescript can **guess the type** of your variable

RULE 2 – If no type, the type is: **any**

RULE 3 – You can specify the type using **annotation** *age : number*

RULE 4 – You can decide a parameter is **optional** *function test(name ? : string)*

RULE 5 – **Inference** of types : Typescript can guess type even of multiple function calls

WANT TO GO FURTHER ?

BASICS TO START TYPESCRIPT

https://www.typescriptlang.org/docs/handbook/2/basic-types.html