TypeScript

PRESENTED BY GROUP-
PRASUK JAIN - IT66
RAJ VERMA - IT71
TANYA SHRIVAS - IT88

What is TypeScript?

- Launched in 2012
- Developed and maintained by **Microsoft**
- **Superset** of JavaScript
- Provides **static typing**

About Types

- The kind or type of information a given program stores
- In JavaScript, there are **six basic data types** which can be divided into **three main categories**:

01. **Primitive data types**
String, Number, and Boolean

02. **Composite data types**
Object, Array, and Function

03. **Special data types**
Undefined and Null

Types

Strings

represent textual data

Undefined

variable declared, but no value assigned

Objects

store collection of data
key-value pair

Arrays

stores multiple values in a single variable

Numbers

positive or negative numbers with or without decimal place

Booleans

2 values - true or false

Null

no value

Functions

bject that executes a block of code

What's the Deal with Types and JavaScript?

- JavaScript is a loosely typed and dynamic language.

```
let foo = 42; // foo is now a number
foo = "bar";  // foo is now a string
foo = true;   // foo is now a boolean
```

when building huge applications, dynamic types can lead to silly bugs in the code base

```
const personDescription = (name, city, age) =>
  `${name} lives in ${city}. he's ${age}. In 10 years he'll be ${age + 10}`;
```

```
console.log(personDescription("Germán", "Buenos Aires", 29));
// Output: Germán lives in Buenos Aires. he's 29. In 10 years he'll be 39.
```

But if accidentally we pass the function the third parameter as a string, we get a wrong output

```
console.log(personDescription("Germán", "Buenos Aires", "29"));
// output: Germán lives in Buenos Aires. he's 29. In 10 years he'll be **2910**.
```

JavaScript doesn't show an error because the program doesn't have a way of knowing what type of data the function should receive.

In Comes TypeScript

- In TypeScript, much like in other programming languages such as Java or C#, we need to declare a data type whenever we create a data structure.

- If there's a match, the program runs, and if not, we get an error.And these errors are very valuable, because as developers we can catch bugs earlier.

In JavaScript

```
const personDescription = (name, city, age) =>
  `${name} lives in ${city}. he's ${age}. In 10 years he'll be ${age + 10}`;
```

In TypeScript

```
const personDescription = (name: string, city: string, age: number) =>
  `${name} lives in ${city}. he's ${age}. In 10 years he'll be ${age + 10}.`;
```

Calling the function with the wrong parameter data type

```
console.log(personDescription("Germán", "Buenos Aires", "29"));
```

// Error: TSError: × Unable to compile TypeScript: Argument of type 'string' is not assignable to parameter of type 'number'.

TypeScript Basics

Types by Inference

In type by **inference**, you don't declare a type at all, but TypeScript infers (guesses) it for you.

```
let helloWorld = "Hello World";
```

If we now try to reassign it to a number, we'll get the following error:

```
helloWorld = 20;
// Type 'number' is not assignable to type 'string'.ts(2322)
```

Declaring Types

The syntax to declare types is quite simple: you just add a colon and its type to the right of whatever you're declaring.

```
let myName: string = "Germán";
```

If we try to reassign it to a number, we'll get the following error:

```
myName = 36; // Error: Type 'number' is not assignable to type 'string'.
```

Interfaces

- An interface looks a lot like a JavaScript object – but we use the interface keyword, we don't have an equal sign or commas, and besides each key we have its data type instead of its value.

```
interface myData {
  name: string;
  city: string;
  age: number;
}

let myData: myData = {
  name: "Germán",
  city: "Buenos Aires",
  age: 29
};
```

Say again I pass the age as a string, I'll get the following error:

```
let myData: myData = {
  name: "Germán",
  city: "Buenos Aires",
  age: "29" // Output: Type 'string' is not assignable to type 'number'.
};
```

Conditionals

If we wanted to make a key conditional, allowing it to be present or not,
we just need to add a question mark at the end of the key in the interface:

```
interface myData {
  name: string;
  city: string;
  age?: number;
}
```

Unions

If we want a variable to be able to be assigned more than one different data type, we can declare so by using **unions** like this:

```
interface myData {
  name: string;
  city: string;
  age: number | string;
}

let myData: myData = {
  name: "Germán",
  city: "Buenos Aires",
  age: "29" // I get no error now
};
```

Typing Functions

When typing functions, we can type its parameters as well as its return value:

```
interface myData {
  name: string;
  city: string;
  age: number;
  printMsg: (message: string) => string;
}

let myData: myData = {
  name: "Germán",
  city: "Buenos Aires",
  age: 29,
  printMsg: (message) => message
};

console.log(myData.printMsg("Hola!"));
```

Typing Arrays

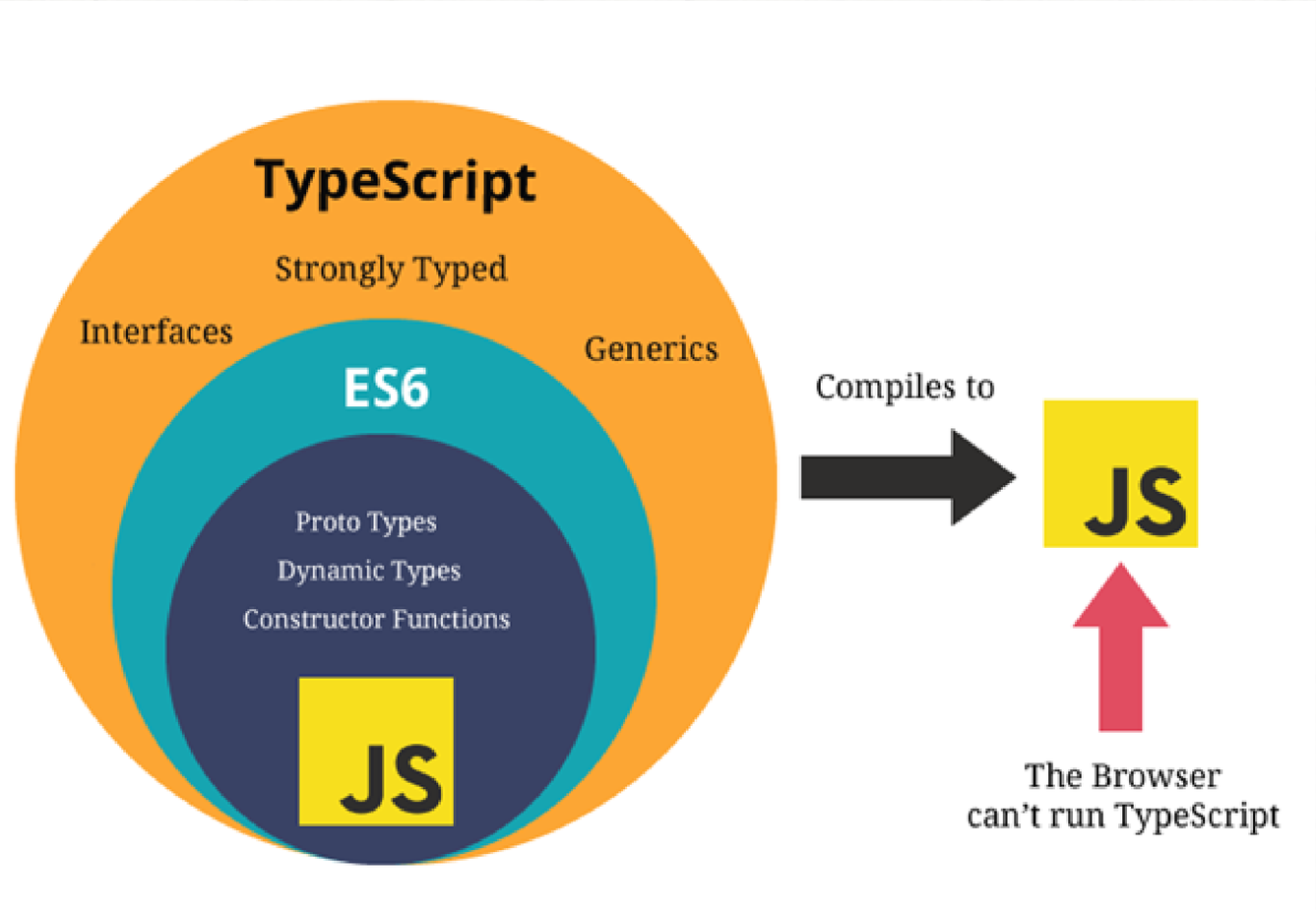
For typing arrays the syntax is the following:

```
let numbersArray: number[] = [1, 2, 3]; // We only accept numbers in this array
let numbersAndStringsArray: (number | string)[] = [1, "two", 3]; // Here we accept numbers and strings.
```

Tuples are arrays with fixed size and types for each position. They can be built like this:

```
let skill: [string, number];
skill = ["Programming", 5];
```

TypeScript's Compiler



- Every time we run our TypeScript file, TypeScript compiles our code and at that point, it checks the types. Only if everything is ok does the program run. That's why we can get errors detected before program execution.

- TypeScript **transpiles** code into JavaScript.

- Browsers don't read TypeScript, but they can execute TypeScript-written programs because the code is converted to JavaScript at build time.

- We can also select to what "flavor" of JavaScript we want to transpile to, for example es4, es5, and so on.

How to Setup a TypeScript Project

- We'll need Node and NPM installed in our system.
- Once we're in the directory of our project, we first run **npm i typescript --save-dev**.
- Then we run **npx tsc --init**. This will initialize your project by creating a **tsconfig.json** file in your directory

We can then create a file with the **index.ts** extension and start writing our TypeScript code. Whenever we need to transpile our code to vanilla JS, we can do it by running **tsc <name of the file>.**

```
const personDescription = (name: string, city: string, age: number) =>
  `${name} lives in ${city}. he's ${age}. In 10 years he'll be ${age + 10}.`;
```

After running **tsc index.ts**, a new **index.js** file is automatically created in the same directory with the following content:

```
var personDescription = function (name, city, age) { return name + " lives in " +
                                                    city + ". he's " + age + ". In 10 years he'll be " + (age + 10) + "."; };
```

Is TypeScript worth it ?

- TypeScript can catch 15% of common bugs, improving code reliability.
- Enhanced code readability in teams makes it easier to understand and collaborate.
- Proficiency in TypeScript expands job opportunities due to its popularity.
- Learning TypeScript provides a deeper understanding of JavaScript and broadens your skill set.

Pros

- Static type-checking
- Improved scalability
- ES6/ES7 support
- Improved IntelliSense support
- Better error handling

Cons

- Stiff learning curve
- Increased complexity
- Slower compilation speeds
- Limited browser support

Thank you very much!