TypeScript

Presentation by:
Filip Markoski 161528
Kristofer Mladenovski 161530

Introduction

- JavaScript is the most popular programming language by far, however it is also the most notorious for being quirky and strange.
- Because of that reason, languages like TypeScript, CoffeeScript and many others try to amend the pitfalls of JavaScript.
- In this project we will explain how TypeScript works and deals with JavaScript’s aforementioned quirkiness

About Transpiling

- TypeScript is transpiled as opposed to compiled.
- Compiling means translating code into a lower level language, whereas transpiling means translating one language into another language in the same level of abstraction
- Whenever you write a TypeScript file, it automatically creates a file of the same name, but with the extension .js, which means that the original TypeScript file gets transpiled into JavaScript

Why use TypeScript?

- The biggest selling point of TypeScript is tooling. It provides advanced autocompletion, navigation, and refactoring.
- TypeScript, as its name suggests, has specific types, which allow for easier debugging and consistency, however the types are completely optional.
- TypeScript has interfaces, which are missing in JavaScript.
- TypeScript has more intuitive object-oriented elements.

Why use types?

- The term dynamic typing refers to the practice where a programmer needn't declare the type of a variable when programming his code, because the variable will dynamically be assigned a type when needed.
- Static typing is the practice where the programmer is forced into declaring the type of every single variable.
- Static typing most of the time leads the programmer to think along the lines of software structure, meaning he needs to understand what variable and more specifically what kind of variable can use a certain function and which cannot.

Basic Types in TypeScript

- Boolean (true or false)
- Number (decimal, hexadecimal, binary and octal floating point numbers)
- String (any textual data)
- Array (a list of items)
- Tuple (arrays with fixed number of elements)
- Enum (enumerator, array of numeric values which have specific names)
- Any (can hold any type)
- Void (opposite of any, denotes absence of any type)
- Null/Undefined (you cannot assign anything to these types)
- Never (denotes values that never occur)

Installation Process

- TypeScript is very easy, straightforward and convenient to install.
- There are 2 main ways to get the TypeScript tools:

1. With npm(Node.js package manager)
2. Installing TypeScript’s Visual Studio plugins

Installing TypeScript with npm

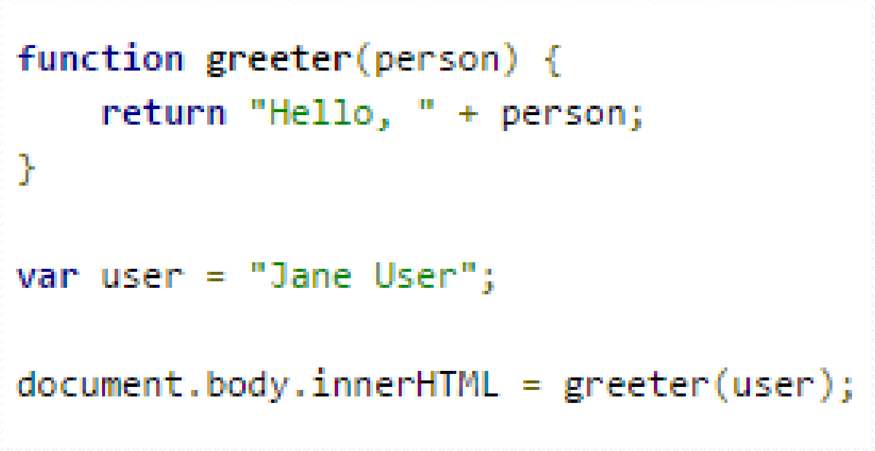
- To install TypeScript with npm you will need to have NodeJS installed with npm, then you can verify the installation by typing “node -v”.
- After the installation has been verified you can type “npm install -g typescript” to install TypeScript itself, the argument “g” meaning global, as in TypeScript will be installed on the whole machine.

Installing TypeScript with Visual Studio

- Visual Studio 2017 and Visual Studio 2015 Update 3 include TypeScript by default.
- And if you didn't install TypeScript with Visual Studio, you can simply download it as a plugin.
- Many other IDEs like WebStorm and PhpStorm also have TypeScript built-in, so you can just start coding in TypeScript if you have any of these IDEs.

Making your first TypeScript file

- Assuming you've went through with the installation process, using the editor of your choice type in the following JavaScript code in a file named "greeter.ts", notice that the extensions is ".ts" instead of ".js".



```
function greeter(person) {
    return "Hello, " + person;
}

var user = "Jane User";

document.body.innerHTML = greeter(user);
```

Making your first TypeScript file

- At the command line, run the TypeScript compiler: "tsc greeter.ts". The result will be a file greeter.js which contains the same JavaScript that we fed in.
- If you want to combine multiple files, you use the line:

  " tsc app.ts another.ts someMore.ts". This will result in files, app.js, another.js and someMore.js.
- If you want to compile all the typescript files in the current folder, use the command: "tsc *.ts"

Keywords and Features in TypeScript

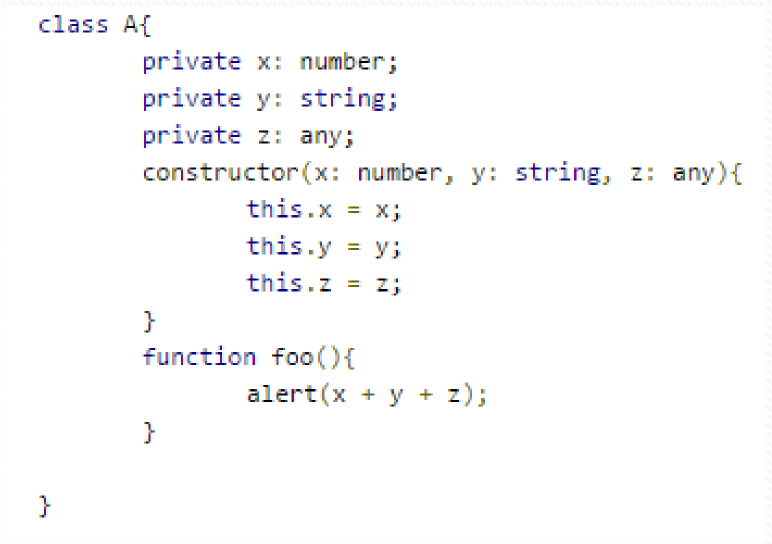
- let
  JavaScript variables are function scoped, and the let variable lets you define variables with true block scopes.
- const
  const allows you to be immutable with variables, i.e. create a variable with a value that doesn't change
- for...of
  for...in in JavaScript does not cycle through items, instead it cycles through the keys of the object passed in, however in TypeScript, for...in correctly iterates over the items.

Keywords and Features in TypeScript

- Multiline Strings
  In JavaScript, creating multiline string was achieved with the escape character ‘\’ and putting a newline manually in the string ‘\n’. In TypeScript, you only need to surround the string with “` `”.
- Arrow Functions
  Arrow functions provide an alternate way to create functions, instead of having to write function all the time you only need to write ()=>something

Object-Oriented Programming in Typescript

- TypeScript allows for a much simple Object-oriented style of programming as opposed to JavaScript because it doesn’t use the strange syntax of prototyping in JavaScript. In TypeScript defining classes is immediately familiar and straightforward as you can see from the following code example:

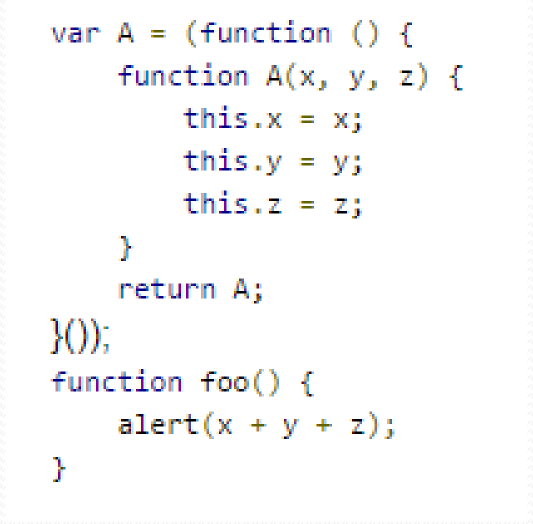


```
class A{
        private x: number;
        private y: string;
        private z: any;
        constructor(x: number, y: string, z: any){
                this.x = x;
                this.y = y;
                this.z = z;
        }
        function foo(){
                alert(x + y + z);
        }

}
```

Object-Oriented programming in TypeScript

- In contrast, the same code in the previous slide, but in JavaScript, would be translated as:

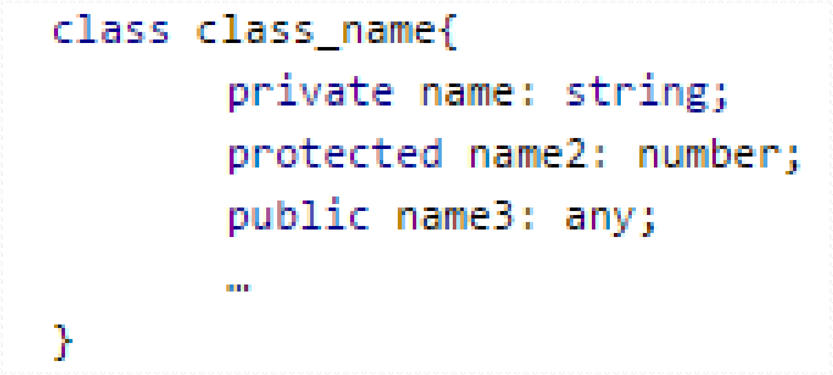


```
var A = (function () {
    function A(x, y, z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }
    return A;
})();
function foo() {
    alert(x + y + z);
}
```

- As we can see, JavaScript creates classes using various functions, and does so in a fairly roundabout way because of its functional nature and the fact that JavaScript was never meant to be an Object-Oriented programming language.

Object-Oriented programming in TypeScript

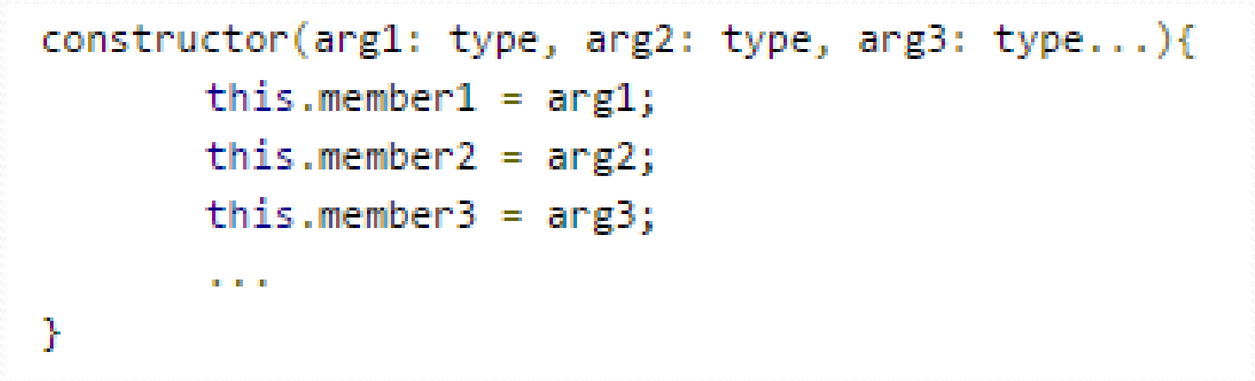
- Creating members of a certain class in TypeScript is pretty simple, we need only to specify whether it's private, protected or public, then typing the name of the member, and finally, specify the type of the member preceded by colons. For example:



```
class class_name{
        private name: string;
        protected name2: number;
        public name3: any;
        …
}
```

Object-Oriented programming in TypeScript

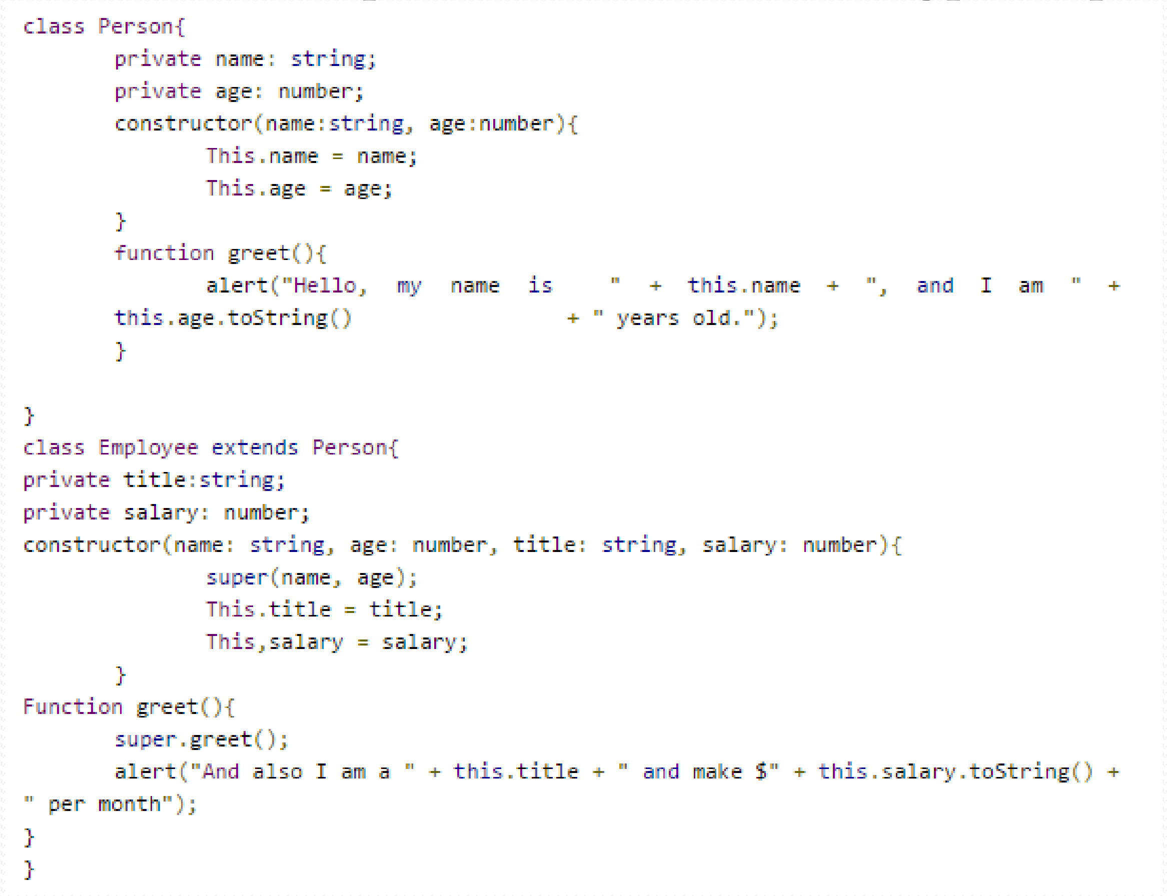
- The syntax for a constructor in TypeScript is pretty straightforward and succinct, as we can see from the following example:



```
constructor(arg1: type, arg2: type, arg3: type...){
        this.member1 = arg1;
        this.member2 = arg2;
        this.member3 = arg3;
        ...
}
```

Object-Oriented programming in TypeScript

- Here is an example of inheritance with TypeScript:

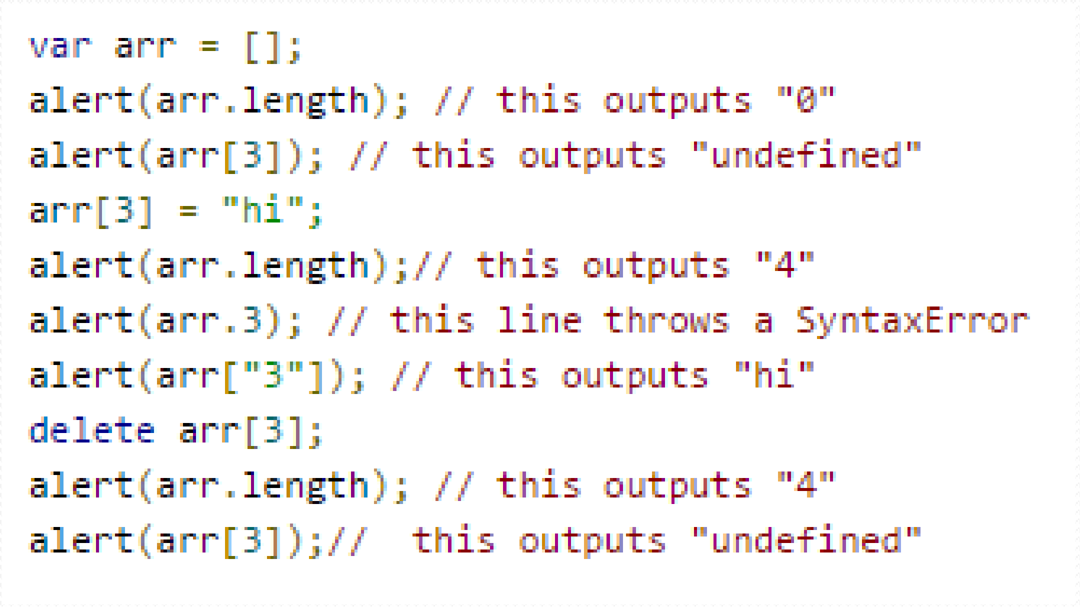


```
class Person{
        private name: string;
        private age: number;
        constructor(name:string, age:number){
                This.name = name;
                This.age = age;
        }
        function greet(){
                alert("Hello,  my  name  is    "  +  this.name  +  ",  and  I  am  "  +
        this.age.toString()                  + " years old.");
        }

}
class Employee extends Person{
private title:string;
private salary: number;
constructor(name: string, age: number, title: string, salary: number){
                super(name, age);
                This.title = title;
                This,salary = salary;
        }
Function greet(){
        super.greet();
        alert("And also I am a " + this.title + " and make $" + this.salary.toString() +
" per month");
}
}
```

Pitfalls of JavaScript

- Array Type:
  Lets consider the following code



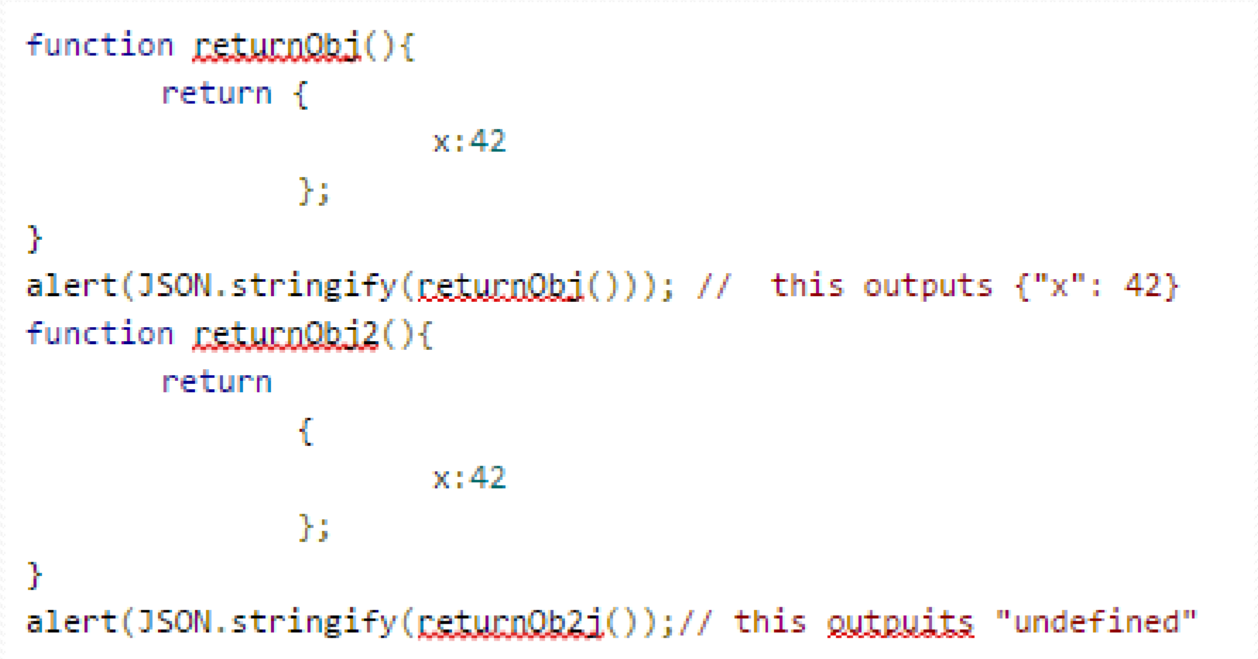
```
var arr = [];
alert(arr.length); // this outputs "0"
alert(arr[3]); // this outputs "undefined"
arr[3] = "hi";
alert(arr.length);// this outputs "4"
alert(arr.3); // this line throws a SyntaxError
alert(arr["3"]); // this outputs "hi"
delete arr[3];
alert(arr.length); // this outputs "4"
alert(arr[3]);//  this outputs "undefined"
```

Pitfalls of JavaScript

- From the code in the previous slide and the results of it we can conclude that arrays in JavaScript have very strange behavior, they behave like a list, sometimes like an object, and sometimes like an array.
- This type of "looseness" isn't isolated in JavaScript, instead it is present throughout the entire language

Pitfalls of JavaScript

- Another example of JavaScript's "looseness" is a feature called optional semicolons, consider the following code:



```
function returnObj(){
        return {
                        x:42
                };
}
alert(JSON.stringify(returnObj())); //  this outputs {"x": 42}
function returnObj2(){
        return
                {
                        x:42
                };
}
alert(JSON.stringify(returnOb2j());// this outpuits "undefined"
```

Pitfalls of JavaScript

- As we can see from the code, the only difference between the two functions is that in the second functions there is a newline after the return statement, yet the first function works while the second gives an error.
- This is because JavaScripts has seemingly random rules as to when it inserts a semicolon, one of those rules is that it always inserts a semicolon after a return statement, causing the previous strange situation.

Conclusion

- From all the information in this presentation, we may conclude that every web developer must learn JavaScript because of its ubiquity in the web development world, however since JavaScript was never meant to create big projects, it has many pitfalls associated with it.
- TypeScript tries to rectify the problems with JavaScript while still reaping its benefits, it has help a lot of programmers with coding efficient and stable websites.