# Typescript

# Agenda

- What's Typescript?
- Type Annotation
- Functions
- Class
- Inheritance
- Module
- Interface
- Generics
- Declaration source files (*.d.ts)
- New features

# What’s Typescript?

- JavaScript is not originally designed for large complex applications (mostly a scripting language, with functional programming constructs), lacks structuring mechanisms like Class, Module, Interface.
- Typescript is a typed superset of JavaScript that compiles to plain JavaScript.
- Adds additional features like Static Type (optional), Class, Module etc. to JavaScript
- Microsoft technology.
- Open Source.
- Versions.
  - First made public in October 2012.
  - Latest version - Typescript 1.7.

# Type Annotation

- Any
  - Any Type is a super set of all types
    - var x : any;
    - var y;
- Primitive
  - Number
    - Does not have separate integer and float/double type.
    - var num : number = 20;
    - var num = 20;
  - String
    - Both single quote or double quotes can be used.
    - var name : string = “hello”;
    - var name =’hello’;
  - Bool
    - var isOpen =true;

# Type Annotation

- Void
  - Used as the return type of functions that don’t return any value

- Object Types
  - class, interface, module.

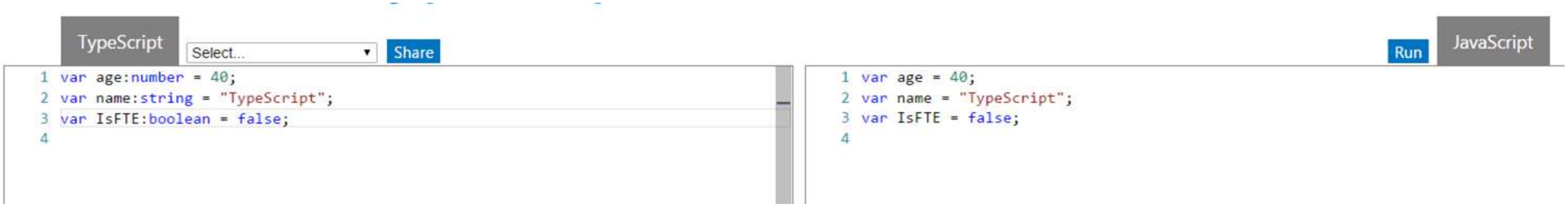
- Array
  - Array types can be written in:
    - var list: number[] = [1, 2, 3];
    - var list: Array<number> = [1, 2, 3];
    - var list:any[] = [1, true, "free"]
- Enum
    - enum Color { Red, Green, Blue };
    - var color = Color.Blue;

# Type Annotation

- Tuple
  - Tuple types allow you to express an array where the type of a fixed number of elements is known.
    - var x: [string, number];
      x = ['hello', 10];

# Type Annotation

- Design time feature. No additional code is emitted in the final JavaScript that TypeScript compiler produces.



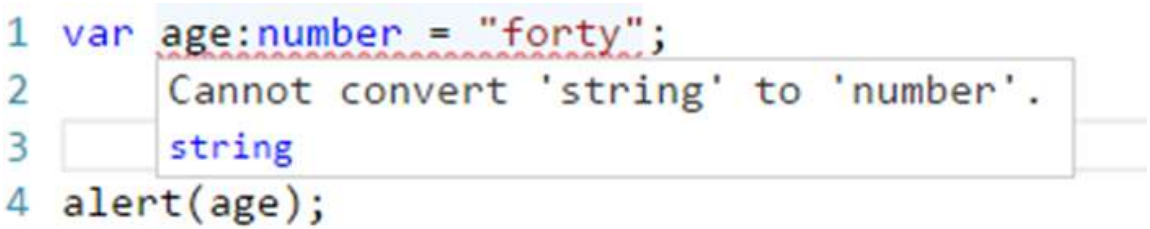
```
var age:number = 40;
var name:string = "TypeScript";
var IsFTE:boolean = false;

```

```
var age = 40;
var name = "TypeScript";
var IsFTE = false;

```

- If there’s a type mismatch TypeScript shows a warning.



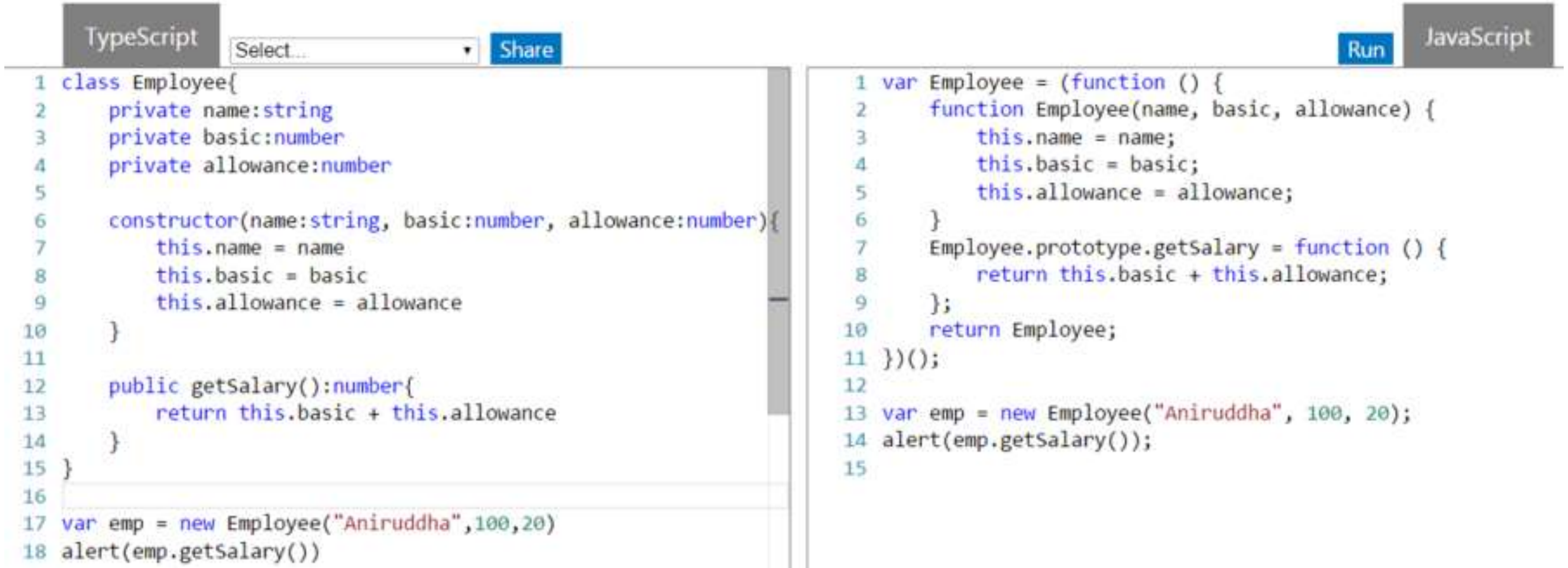
```
var age:number = "forty";
       Cannot convert 'string' to 'number'.
       string
alert(age);
```

# Functions

- Type Annotation for parameter and return type.
- Optional and Default Parameter.
- Function Overloads.
- Fat Arrow functions.
- Rest parameters.
    - denoted by ‘...argumentName’ for the last argument allow you to quickly accept multiple arguments in your function and get them as an array.

# Class

- Properties and fields to store data
- Methods to define behavior



```typescript
class Employee{
    private name:string
    private basic:number
    private allowance:number

    constructor(name:string, basic:number, allowance:number){
        this.name = name
        this.basic = basic
        this.allowance = allowance
    }

    public getSalary():number{
        return this.basic + this.allowance
    }
}

var emp = new Employee("Aniruddha",100,20)
alert(emp.getSalary())
```

```javascript
var Employee = (function () {
    function Employee(name, basic, allowance) {
        this.name = name;
        this.basic = basic;
        this.allowance = allowance;
    }
    Employee.prototype.getSalary = function () {
        return this.basic + this.allowance;
    };
    return Employee;
})();

var emp = new Employee("Aniruddha", 100, 20);
alert(emp.getSalary());
```

# Inheritance

- Typescript supports inheritance of class through extends keyword
- super keyword.

# Module

- Modules can be defined using module keyword.
- A module can contain sub module, class, interface or enum.
- Class, Interfaces , functions can be exposed using export keyword.

- Adding file references.  - /// <reference path=“filename.ts" />

# Interface

- Declared using interface keyword
- TS compiler shows error when Interface signature and implementation does not match
- Optional properties can be declared for an interface (using ?)

# Generics

- Able to create a component that can work over a variety of types rather than a single one.

```
function identity<T>(arg: T): T {
        return arg;
}
```

- type argument inference - we want the compiler to set the value of T for us automatically based on the type of the argument we pass in.

# Declaration source files (*.d.ts)

- The purpose of these files is to provide some typing information for JavaScript libraries.
- These files do not compile to .js files, simply describe their JavaScript implementations, and act as their representative.
- Contains typing info.

- http://definitelytyped.org/

# New features

- await & async
- for..of – iteration.
- Exponentiation operators -  ** and **=
- Spread operator