Typescript Installation Instructions

Installation Guide

Installation Guide

INTRODUCTION

1. TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.
2. TypeScript allows providing types for primitive data types.
3. Typescript provides Classes, Modules, and Interfaces.

INSTALLATION

1. TypeScript needs NodeJS installed.
2. TypeScript can be installed using the command: *npm install -g typescript*
3. TypeScript files can be compiled to your target JS Version using the compiler. The compiler can be used from the shell: *tsc helloworld.ts*

TYPESCRIPT USAGE

CLASSES AND DATA TYPES

1. Classes can have definitions as well as assignations
2. Assigned types have to be of the same type as assigned to the variable
3. You can assign public/private modifiers for methods

```
class Point{
        x: number; // Assigning types to primitive data types
        y: string;
        a: number[]; // Assigning Array Types
        pointer: Pointer;
        constructor(x: number, y: string, pointer: Pointer) {
                this.x = x; // Assigning value to primitive data types with same data types
                this.y = y;
                this.pointer = pointer; // Assigning interface of same data type to variable
        }
        // Assigning public / private modifiers
```

```
        private getDist(){
                return Math.sqrt(this.x * this.x + this.y * this.y);
        }
        public getXValue(){
                return x;
        }
}
var p = new Point(3,4,{point:"PointString"});
```

INTERFACES

1. Interfaces are used for definition or type structure for objects and do not have values or assignation or more precisely no definition
2. Interfaces do not have footprint on the compiled JS files

```
// Creating a pointer
interface Pointer{
    point: string;
}
```


INTERFACE IMPLEMENTATION

1. Use the implements syntax to implement an interface

```
// Implementing the interface
class GeoPointer implements Pointer{
  point: string;
  constructor(gp: string) {
    this.point = gp;
  }
}
```

EXTENDING THE INTERFACE

1. Use the *extend* syntax to extend an interface or class

```
// Extending an interface or class
interface Squareclock extends GeoPointer{
   sideLength: number; // Other values in the class other than GeoPointer
   hvalue: number;
   mvalue: number;
   constructor(h: number, m: number) {
      this.hvalue = h;
      this.mvalue = m;
   }
}
```

ENUM

1. An *enum* is a way of giving more friendly names to sets of numeric values

```
// Enumeration definitions
enum Color {Red, Green = 1, Blue};
var colorName: string = Color[2];
```