**TypeScript**

TypeScript is a typed superset of JavaScript that compiles to plain JavaScript.

**Installation:**

To install TypeScript we can do it with **npm** and **Visual Studio**:

**npm**

**install**

npm install -g typescript

**compile**

tsc helloworld.ts

**Visual Studio**

<https://www.typescriptlang.org/#download-links>

**Basic Types:**

boolean:  is the simple true/false value.

let isDone: boolean = false;

number:  all numbers in TypeScript are floating point values.

let decimal: number = 6;

let hex: number = 0xf00d;

let binary: number = 0b1010;

let octal: number = 0o744;

string:

let color: string = "blue";

color = 'red';

array: arrays of values

let list: number[] = [1, 2, 3];

tuple:

let x: [string, number];

x = ["hello", 10];

enum:  is a way of giving more friendly names to sets of numeric values.

enum Color {Red, Green, Blue}

let c: Color = Color.Green;

any: describe the type of variables that we do not know when we are writing an application.

let notSure: any = 4;

notSure = "maybe a string instead";

notSure = false;

void: is a little like the opposite of any, we may commonly see this as the return type of functions that do not return a value:

**null** and **undefined.**

never.

object.

**Variables**

Is preferable use **let** and **const** instead of **var.**

**Interface**

A interface is a object that define a behavior of a class that implements it.

interface ClockInterface {

currentTime: Date;

}

class Clock implements ClockInterface {

currentTime: Date = new Date();

constructor(h: number, m: number) { }

}

We can use a Interface like a class.

**Classes**

A class is a object that has properties(attributes) and behaviors(functions).

class Greeter {

greeting: string;

constructor(message: string) {

this.greeting = message;

}

greet() {

return "Hello, " + this.greeting;

}

}

let greeter = new Greeter("world");

**Ineheritance**

Is a way to extend properties and behaviors to others classes.

class Animal {

move(distanceInMeters: number = 0) {

console.log(`Animal moved ${distanceInMeters}m.`);

}

}

class Dog extends Animal {

bark() {

console.log('Woof! Woof!');

}

}

const dog = new Dog();

dog.bark();

dog.move(10);

**Public, private and protect modifiers**

In TypeScript all variables, functions and sentences are public.

When a member is marked as **public** modifier can be accessed from outside classes.

When a member is marked as **private**, it cannot be accessed from outside of its containing class.

When a member is marked as **protected**, it can be accessed from his deriving classes.

When a property is marked as **readonly**, it can be access only for read.

When a property is marked as **static**, it can be access only into the class.

**Abstract classes**

An abstract class is like a interface but with behavior

**Functions**

In typescript all parameters that recives a functions is required, but we can use a default(=) or

Optional(?) parameter.

function buildName(firstName: string, lastName?: string) {

// ...

}

function buildName(firstName: string, lastName = "Smith") {

// ...

}

We can use arrow functions too.

**Generics**

We can use generics data types, using <Type>

function identity<T>(arg: T): T {

return arg;

}

**Iterators**

**for … on** returns a list of values

**for … in** returns a list of key

let list = [4, 5, 6];

for (let i in list) {

console.log(i); // "0", "1", "2",

}

for (let i of list) {

console.log(i); // "4", "5", "6"

}

**Modules**

We can export our modules to can be used in other files with **export** and we can use the

modules with **import,** these features works same to **require()** of JavaScript.

**Documentation**:

<https://www.typescriptlang.org/docs/home.html>