VERSION 1.0

Guía No. 1. Typescript desde Cero

Objetivos.

- Al finalizar todos los participantes tendrán una idea clara idea de lo que es Typescript.
- Explicar los tipos básicos.
- Hacer el primer programa “Hola mundo” de la comunidad.

Requisitos:

- Deseo de aprender constantemente y de aprender haciendo.
- Tener instalado el un entorno de desarrollo basado en Linux. (VsCode, Git, Curl).

Algunas Consideraciones Iniciales:

A la fecha de hoy Septiembre de 2018 TypeScript está en la versión 3.0 y trae consigo nuevas características que se irán estrenando poco a poco en la comunidad a medida que los programas lo demanden.

Por el momento vamos a mencionar algunas cosas sobre el lenguaje como tal. Lo primero y lo más importante mencionar es que TypeScript es de **código abierto** bajo la licencia de Apache 2.0 y apoyado 100% por microsoft, es decir que lo podemos usar

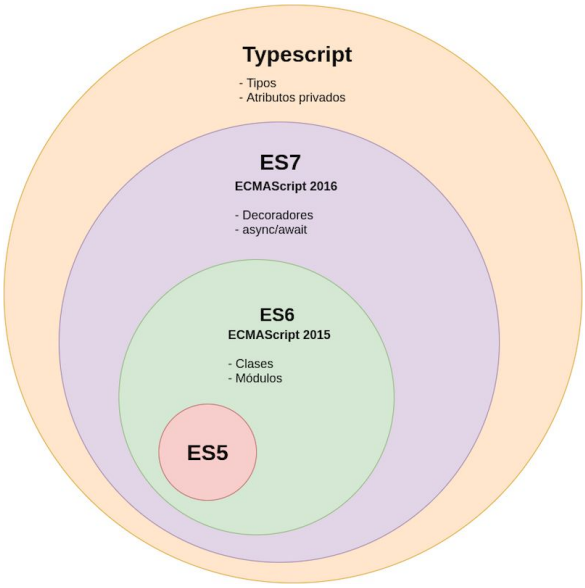


Figura 1. TypeScript como superconjunto de JavaScript. Tomado de 2

sin temor a infringir derechos de autor ni pagar costosas licencias, otra cosa no menos importante es mencionar que TypeScript **NO ES** una librería y mucho menos un Framework de JavaScript, TypeScript **ES** un superconjunto de JavaScript (no tiene nada que ver con Java), o lo que es lo mismo pero dicho en otras palabras, JavaScript es un subconjunto de TypeScript, esto quiere decir que todo el código escrito en TypeScript se va a traducir directamente a JavaScript en cualquiera de sus versiones estables ES3-ES6 siendo este último es el que se va a ejecutar en el navegador, si esto es así, algunos se preguntarán porqué no simplemente escribir el código directamente en JavaScript ??

La respuesta es concretamente porque TypeScript es un lenguaje moderno y completo desde su concepción y diseño que permite organizar mejor el código, además ofrece a los desarrolladores desde ya características que no se van a ver directamente en JS si no hasta algún punto desconocido y quizá distante en el futuro.

Dentro de sus grandes ventajas encontramos el sistema de anotaciones de tipo que ofrece a los desarrolladores la opción de detectar errores sintácticos o semánticos mientras desarrollamos la idea de nuestro programa, esto es en antes de ejecutar el código en el navegador ya estamos detectando posibles fallos y de esta manera estamos reduciendo al máximo fallos de funcionalidad de nuestro programa, pero sin utilizar aumentar la complejidad del código base utilizando librerías externas.

LO QUE VAMOS A HACER HOY.

Actividad Inicial.

En el día de hoy vamos a revisar las estructuras de almacenamiento básicas de TypeScript, pero antes revisemos las aplicaciones que vamos a utilizar, en caso de no tenerlas instaladas podemos hacer uso de esta guía para instalarlas. Continuemos ¡¡.

1. Instalar el aplicativo de linux Curl. Abrimos una sesión de terminal y escribimos la línea, con permisos de super usuario.

```
>_  sudo apt install curl
```

Curl es una aplicación que se ejecuta directamente en la terminal (no tiene interfaz gráfica), y se utiliza para hacer peticiones HTTP a un servidor Web y recibir su

respuesta como información cruda. Como se pueden dar cuenta es una aplicación bastante útil y que vamos a aprender a utilizar en futuros encuentros.
Más información aquí. https://linuxhint.com/install-curl-on-ubuntu-18-04/

2. Instalar git.

Para instalar Git primero vamos a agregar el repositorio oficial de Git para Ubuntu, para esto abrimos una sesión en el terminal y agregamos las siguientes operaciones con permisos de super usuario.

>_ **sudo add-apt-repository** ppa:git-core/ppa

Actualizamos el repositorio

>_ **sudo apt update**;

Finalmente instalamos con seguridad Git y todas sus dependencias.

>_ **apt install** git

Git es un programa que nos va a yudar a llevar un registro de todos los cambios que se vayan dando en nuestros programas, más adelante hablaremos mucho más de Git por el momento sólo lo vamos a instalar en nuestros equipos.

3. Instalar node JS con el manejador de versiones de Node NVM, Seleccionar la última versión v8.12.0.

1. Primero instalamos el manejador de versiones de node con la aplicación Curl, copiamos en el terminal el siguiente comando.

>_ curl -o- https://raw.githubusercontent.com/creationix/nvm/v0.33.11/install.sh | bash

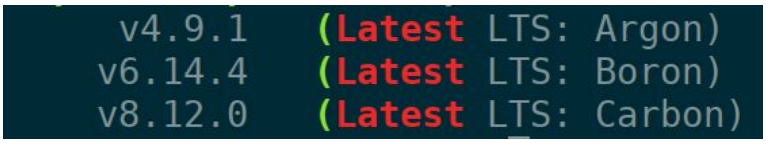
Verificamos la instalación con la operación .. nvm --versión
Procedimiento sugerido en: https://github.com/creationix/nvm

2. Guardamos la ruta en el PATH con la operación

>_ **source** ~/.profile

3. Listamos las últimas versiones de node para utilizar

>_ nvm ls-remote | grep Latest



Instalamos la opción LTS Carbon, en este caso la V8.12.0

>_ nvm install 8.12.0 (Ultima versión a la fecha hoy)

Después de instalar la Versión 8.12.0 de node, seleccionar la siguiente opción en el terminal.

>_ nvm use 8.12.0

Verificar la opción seleccionada con la siguiente línea.

>_ node -v (v minúscula)

La salida debe ser la versión que acabamos de instalar **8.12.0**

4. Instalar yarn administrador de paquetes,

1. Primero agregamos el repositorio oficial de yarn.

>_ **curl** -sS https://dl.yarnpkg.com/debian/pubkey.gpg | sudo apt-key add -

>_ **echo** "deb https://dl.yarnpkg.com/debian/ stable main" | sudo tee /etc/apt/sources.list.d/yarn.list

2. Actualizamos todos los repositorios e instalamos yarn en su última versión.

>_ **sudo** apt-get update && sudo apt-get install yarn

3. Agregar la ruta de los binarios instalados con yarn al PATH.

Abrimos el archivo de perfiles con el editor de texto gedit, con el siguiente comando

 **gedit** ~/.profile.

Al final del archivo que acabamos de abrir Agregamos la siguiente línea.

export PATH="$HOME/.yarn/bin:$PATH"

Agregando esta línea en este archivo de configuración, linux ya va a reconocer la ruta para llegar al compilador de typeScript y en general de todos los binarios que se instalen con la opción global.
Procedimiento adaptado de:
https://unix.stackexchange.com/questions/26047/how-to-correctly-add-a-path-to-path#26059

4. Guardamos los cambios en el archivo y se lo hacemos saber al Sistema Operativo, con el siguiente comando.

>_ **source** ~/.profile

5. Instalar typescript en los equipos o máquina virtual.

>_ sudo yarn **global** add typescript

Typescript hace parte de los más de 5000 paquetes del repositorio de npm, y con este comando se van a instalar dos pequeños programas **tsc** y **tsserver** que son el compilador ó mejor transpilador de typescript y el servidor respectivamente, este último lo utiliza el IDE para incorporar typescript dentro de su entorno como archivo de configuración.
Procedimiento sugerido en:
https://dmitryzinchenko.com/2016/08/30/installing-typescript-on-ubuntu-desktop/

Al finalizar Instalamos las siguientes extensiones en VS Code ..
TypeScript Extension Packt, Prettier, Bracket Pair Colorizer, Live Server, Path IntelliSense.

Con TypeScript instalado vamos a conocer las estructuras nativas esenciales para escribir código en este lenguaje.

Los Boleanos - tipo *Boolean*

Son muy útiles para evaluar estructuras condicionales o también en ciertos casos para verificar el estado de la aplicación, por ejemplo para verificar que un empleado aún se encuentra con su contrato vigente (**esta_activo** = true) o no (**esta_activo** = falso).

Los valores numéricos - tipo *number*
Se utilizan para hacer operaciones aritméticas, como por ejemplo llevar la cuenta de cuántos usuarios han utilizado la aplicación en la última hora.

Las Cadenas de Caracteres - tipo *String*
Se utilizan para almacenar palabras u oraciones completas, generalmente se utilizan para almacenar información relacionada con el usuario, información tal como nombre, dirección, incluso un número telefónico, a pesar de que en este último caso la información se refiere a caracteres numéricos, es una buena práctica almacenar este tipo de información como cadenas de texto y no como valores numéricos.

Los símbolos - tipo *Symbol*
Es único e inmutable, se utiliza con frecuencia en la definición de las propiedades de un objeto dentro de una clase.

Las colecciones de elementos del mismo tipo - tipo *Arrays* de cualquiera de los tipos anteriores.

Las Interfaces - Colecciones de elementos de diferente tipo que agrupados conforman un nuevo tipo unificado de almacenamiento personalizado.

Enumeraciones *Enums*
Cuando tenemos varias opciones, por ejemplo queremos almacenar las diferentes ciudades de Colombia, esta estructura es una excelente alternativa para almacenar este tipo de información.

ALGUNAS CONSIDERACIONES IMPORTANTES

Con respecto a la definición de una variable es importante no repetir el mismo nombre que ya se ha utilizado para definir una variable anterior.

Por buena práctica vamos a emplear **la secuencia encadenada** (*Snake Case*) para nombrar variables compuestas por 2 o más palabras esto es utilizar un guión bajo como puente de enlace entre dichas palabras y todas escritas en minúsculas, nada de mayúsculas. Ejemplo: *let* temperatura_actual y no TemperaturaActual o Temperatura actual.

Tampoco vamos a utilizar caracteres numéricos iniciando el nombre de una variable, aparte TS te va a generar una alarma de error. Ejemplo *let* 1_temp_1

En este punto ya contamos con las conocimientos básicos para construir nuestro primer programa Hola mundo que consiste en convertir una valor de temperatura de grados Celsius a una temperatura de grados Farenheit, de la misma forma vamos a hacer la conversión de un valor dado inicialmente en grados Farenheit y convertirlo en grados Celsius con base en las siguientes fórmulas de conversión.

De grados Celsius a grados Farenheit

$$C = 5/9 \times (F-32)$$

De grados Farenheit a grados Celsius

$$F = [(9/5)C] + 32$$

Estrategía sugerida para el desarrollo de este programa.

Para este caso vamos inicialmente a definir 3 constantes de tipo numérico, las constantes como su nombre sugiere son valores que después de ser asignados a una variable no cambian en toda la extensión del programa y se definen con la palabra reservada ***const***

// Definición de Constantes

Cuando una expresión comienza con los dos barras frontales (//), indica que es un comentario dentro del programa y no hace parte del orden de ejecución del mismo.

A continuación vamos a definir los valores constantes que se observan en la fórmula

```
const factor_conversion_c = 5/9;
const factor_conversion_f = 9/5;
const factor_cero = 32;
```

Notemos que TypeScript también reconoce valores numéricos expresados como fracciones.

Ahora vamos a definir un valor para la temperatura en grados Farenheit y lo vamos a almacenar en una variable cuyo nombre será (temperatura_actual_f) de la siguiente manera, para esto utilizamos la palabra reservada ***let,*** como se muestra a continuación..

let temperatura_actual_f = 87;

Vamos a definir también una variable para almacenar la temperatura en grados Celsius.

let temperatura_actual_c = 28;

A continuación vamos a realizar las operaciones definidas en la fórmula y la vamos el resultado en dos variables como se muestra a continuación.

// Definir una variable con nombre conversion_a_celsius.

let conversion_a_celsius = factor_conversion_c * (temperatura_actual_f - factor_cero)

// Definir una variable con nombre conversion_a_farenheit.

let conversion_a_farenheit=(factor_converison_f ***** temperatura_actual_c) **+** factor_cero

Finalmente vamos a mostrar los resultados en consola utilizando la función **console.log()**

console.log(“La temperatura actual en grados Farenheit es”, temperatura_actual_f, “y su valor en grados Celsius es”, conversion_a_celsius**)**

console.log(“La temperatura actual en grados Celsius es”, temperatura_actual_c, “y su valor en grados Farenheit es”, conversion_a_farenheit**)**

Finalmente vamos a abrir una sesión de terminal y vamos a correr el compilador de typescript seguido del nombre del archivo .ts

tsc **nombre_del_archivo.ts**, esta línea transpila el código de TypeScript a JavaScript y se genera un nuevo archivo … **nombre_del_archivo.js**

Referencias Bibliográficas.

1. https://www.c-sharpcorner.com/article/getting-started-with-typescript-part-one/
2. https://miguelgomez.io/typescript/typescript-javascript-introduccion/
3. https://ciphertrick.com/2018/05/07/winning-sides-typescript-javascript/