Angular

**INSTALAR ANGULAR/CLI**

Necesitamos instalar **angular/cli** con el comando:

**Npm install –g @angular/cli**

Y luego ejecutar el comando:

**ng versión**

**BASES DE TYPESCRIPT**

**TIPOS BÁSICOS Y CONCEPTOS GENERALES**

Podemos especificarle a la variable que tipo de datos es el valor de la misma:

let nombre: string = "joaquin";

Como también podemos especificar que la variable pueda recibir dos tipos de datos:

let numberAndString: number | string = 95;

numberAndString = "hola";

**OBJETOS, ARREGLOS E INTERFACES**

Debemos especificarle al array que tipos de datos puede recibir:

let habilidades: (boolean | string | number)[] = ["Bash", "Counter", 100]

habilidades.push(true);

Para especificar los tipos de datos que recibe un objeto utilizamos una **interface**, esta interface es una estructura base de lo que tiene que contener el objeto a crear. Para agregar **propiedades** **opcionales** hacemos uso del **signo de interrogación (?)**.

interface Personaje {

    nombre: string,

    hp: number,

    habilidades: string[],

    puebloNatal?: string

}

const personaje: Personaje = {

    nombre: "joaquin",

    hp: 24,

    habilidades: ["Bash", "Counter", "Healing"]

}

personaje.puebloNatal = "Pueblo Paleta";

console.table( personaje );

**FUNCIONES BÁSICAS**

A las funciones le podemos especificar qué tipo de datos reciben y también que tipo de dato esperamos que sea devuelto:

function sumar (a: number, b: number): number {

    return a + b;

}

const resultado = sumar(10, 20);

console.log(resultado)//30

**Arrow function:**

const sumarFlecha = (a: number, b: number):number => {

   return a + b

}

Podemos pasar argumentos opcionales y por default. El orden para colocar bien los argumentos son: requeridos, opcionales y por default.

function multiplicar (a: number, b?: number, c: number = 2) {

    return a \* c

}

const resultadoMultiplicacion = multiplicar(10, 0, 10)

console.log(resultadoMultiplicacion)//100

**FUNCIONES CON OBJETOS COMO ARGUMENTOS**

Si en una function no quiero retornar nada utilizo **void**.

interface PersonajeLOR {

    nombre: string,

    pv: number,

    mostrarPV: () => void

}

function curar (personaje: PersonajeLOR, curarX: number): void {

    personaje.pv += curarX;

    console.log(personaje)

}

const nuevoPersonaje: PersonajeLOR = {

    nombre: "Frodo",

    pv: 50,

    mostrarPV() {

        console.log("Puntos de vida: ", this.pv)

    }

}

curar(nuevoPersonaje, 20);

nuevoPersonaje.mostrarPV();

**TAREA SOBRE OBJETOS E INTERFACES**

interface SuperHero {

    nombre: string,

    edad: number,

    direccion: Direccion,

    mostrarDireccion: () => string

}

interface Direccion {

    calle: string,

    pais: string,

    ciudad: string

}

const superHero = {

    nombre: "Spiderman",

    edad: 30,

    direccion: {

        calle: "Main St",

        pais: "USA",

        ciudad: "NY"

    },

    mostrarDireccion () {

        return `${this.nombre}, ${this.direccion.ciudad}, ${this.direccion.pais}`

    }

}

const direccion = superHero.mostrarDireccion();

console.log(direccion);

**DESTRUCTURACIÓN DE OBJETOS**

interface Reproductor {

    volumen: number,

    segundo: number,

    cancion: string,

    detalles: Detalles,

}

interface Detalles {

    autor: string,

    anio: number

}

const reporductor: Reproductor = {

    volumen: 90,

    segundo: 36,

    cancion: "Mess",

    detalles: {

        autor: "Ed Sheeran",

        anio: 2015

    }

}

const autor = "Fulano";

const {volumen, segundo, cancion, detalles} = reporductor;

const {autor: autorDetalle} = detalles

console.log(`El volumen actual es: ${volumen}`);

console.log(`El segundo actual es: ${segundo}`);

console.log(`La canción actual es: ${cancion}`);

console.log(`El autor actual es: ${autorDetalle}`);

**DESESTRUCTURACIÓN DE ARRAYS**