REQUISITOS PREVIOS

Repaso a Typescript

Parte II

```
/**
 * Any
 */
let notSure: any = 4; // En el momento de inicializar la variable la asignamos a un number
notSure = "Nueva cadena de texto"; // Ahora pasará a ser un string
notSure = false; // Por último es de un tipo boolean

let lista: any[] = [1, true, "Cadena"];
lista[1] = 100;
```

ANY

Utilizaremos este tipo cuando aún no tengamos definida nuestra estructura de datos.

```typescript
/**
 * Clases
 */
class Persona {
    private nombre: string;
    private edad: number;

    constructor(nombre: string, edad: number) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public saludar(): void {
        console.log(`Hola, mi nombre es ${ this.nombre } y tengo ${ this.edad } años.`);
    }
}

let persona = new Persona("Jonatan Lucas", 32);
persona.saludar();
// Hola, mi nombre es Jonatan Lucas y tengo 32 años.
```

CLASES en Typescript

Como un lenguaje de programación orientado a objetos, Typescript permite el uso de clases con sus atributos y métodos.

INTERFAZ en Typescript

```typescript
/**
 * Interfaz
 */
interface Direccion {
    calle: string;
    num: number;
    poblacion: string;
}

/**
 * Clases
 */
class Persona {
    private nombre: string;
    private edad: number;
    private direccion: Direccion;

    constructor(nombre: string, edad: number) {
        this.nombre = nombre;
        this.edad = edad;
    }

    public saludar(): void {
        console.log(`Hola, mi nombre es ${ this.nombre } y tengo ${ this.edad } años.`);
    }

    public definirDireccion(direccion: Direccion): void {
        this.direccion = direccion;
    }
}

let persona = new Persona("Jonatan Lucas", 32);
persona.saludar();
// Hola, mi nombre es Jonatan Lucas y tengo 32 años.

persona.definirDireccion({
    calle: "Discordia",
    num: 3,
    poblacion: "Atlantis"
});
```

Las interfaces, *interface*, se utilizan para crear objetos específicos para definir un objeto de datos.

```ts
/**
 * Funciones
 */
function sumar(x: number, y: number): number {
    return x + y;
}

console.log(sumar(2, 5));
// 7
```

FUNCIONES

En Typescript, las funciones juegan el rol de describir cómo hacer algo, o de cómo realizar una función específica.

```
function construirNombre(nombre: string, apellido?: string): string {
    if (apellido) {
        return nombre + " " + apellido;
    } else {
        return nombre
    }
}

console.log(construirNombre("Jonatan"));
// Jonatan
console.log(construirNombre("Jonatan", "Lucas"));
// Jonatan Lucas
console.log(construirNombre("Jonatan", "Lucas", "Lucas"));
// Error, demasiados parámetros
```

PARÁMETROS OPCIONALES

Para tener parámetros opcionales en una función, o cualquier método, en Typescript se le coloca al final del nombre del parámetro el símbolo '?'