TypeScript

3. Interfaces

1. Introducción

- Con las interfaces, se puede dar un paso más y definir la estructura o tipo de objetos más complejos en el código.
- Al igual que tipos de variables simples, estos objetos también tendrá que seguir un conjunto de reglas creadas por nosotros.
- Esto puede ayudarnos a escribir código con más confianza, con menos posibilidades de error.

2. Creando Nuestra Primera Interfaz

- Supongamos que tenemos un objeto de lago en el código y vamos a usarlo para almacenar información sobre algunos de los lagos más grande por área en el mundo.
- Este objeto de lago tiene propiedades como el nombre del lago, su área, longitud, profundidad y los países en que existe ese lago.

2. Creando Nuestra Primera Interfaz

```
interface Lagos {
    nombre: string,
    area: number,
    longitud: number,
    profundidad: number,
    aguaDulce: boolean,
    paises: string[]
}
```

- La interfaz Lagos contiene el tipo de cada propiedad que vamos a utilizar al crear los objetos de nuestro lago.

2. Creando Nuestra Primera Interfaz

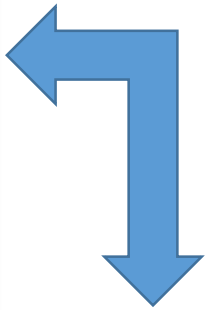
- Si ahora intenta asignar a diferentes tipos de valores a cualquiera de estas propiedades, se producirá un error.
- Ejemplo que almacena información acerca de nuestro primer lago.

```
let primerLago: Lagos = {
    nombre: 'Mar Caspio',
    longitud: 1199,
    profundidad: 1025,
    area: 371000,
    aguaDulce: false,
    paises: ['Kazakhstan', 'Russia', 'Turkmenistan', 'Azerbaijan', 'Iran']
}
```

2. Creando Nuestra Primera Interfaz

- No importa el orden en que asigna un valor a estas propiedades. Sin embargo, no se puede omitir un valor.

```
let segundoLago: Lagos = {
    nombre: 'Superior',
    longitud: 616,
    area: 82100,
    aguaDulce: true,
    paises: ['Canada', 'United States']
}
```



```
let segundoLago: Lagos

La propiedad "profundidad" falta en el tipo "{ nombre: string; longitud: number; area:
number; aguaDulce: true; paises: string[]; }", pero es obligatoria en el tipo
"Lagos". ts(2741)

lagos.ts(5, 5): "profundidad" se declara aquí.

Ver el problema    Corrección Rápida (Ctrl+.)

}

let segundoLago: Lagos = {
```

3. Propiedades de la Interfaz Opcionales

- A veces, puede que necesite una propiedad sólo para algunos objetos específicos.
- Si añadimos un signo de interrogación **(?)** después del nombre de una propiedad, se establece como **opcional** en la declaración de interfaz.

```
interface Lagos {
    nombre: string,
    area: number,
    longitud: number,
    profundidad: number,
    aguaDulce: boolean,
    paises: string[],
    mesesHelado?: string[]
}
```

3. Propiedades de la Interfaz Opcionales

```
let tercerLago: Lagos = {
    nombre: 'Superior',
    profundidad: 406.3,
    longitud: 616,
    area: 82100,
    aguaDulce: true,
    paises: ['Canada', 'United States']
}

let cuartoLago: Lagos = {
    nombre: 'Baikal',
    profundidad: 1637,
    longitud: 636,
    area: 31500,
    aguaDulce: true,
    paises: ['Russia'],
    mesesHelado: ['Enero', 'Febrero', 'Diciembre']
}
```

4. Propiedades de Sólo Lectura

- Cuando se trabaja con objetos diferentes, puede que necesite trabajar con propiedades que sólo pueden modificarse cuando creamos primero el objeto.
- Puede marcar estas propiedades como **readonly** en la declaración de interfaz.
- Esto es similar al uso de la palabra clave **const**.

4. Propiedades de Sólo Lectura

```
interface Enemy {
    readonly size: number,
    health: number,
    range: number,
    readonly damage: number
}
```

```
let tank: Enemy = {
    size: 50,
    health: 100,
    range: 60,
    damage: 12
}


// Okay
tank.health = 95;

// Error porque 'damage' es de solo lectura
tank.damage = 10;
```

```
// Ok
tank.
```

(property) Enemy.damage: any

No se puede asignar a "damage" porque es una propiedad de solo lectura. ts(2540)

Ver el problema No hay correcciones rápidas disponibles

```
// Er
tank.damage = 10;
```