TYPESCRIPT (2)

4. Funciones y Tipos de Retorno:

En TypeScript, se puede especificar los tipos de los parámetros y del valor de retorno de las funciones.

1. Parámetros de la función: se pueden declarar los tipos de los parámetros de la función entre paréntesis después del nombre de la función. Los tipos de los parámetros pueden ser cualquier tipo de datos válido en TypeScript.

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

Para especificar parámetros opcionales se añade un signo de interrogación al final del nombre del parámetro.

2. Valor de retorno de la función: se puede especificar el tipo de dato que devuelve una función colocando : tipo después de los parámetros de la función. El tipo de retorno puede ser cualquier tipo de datos válido en TypeScript. Si una función no devuelve ningún valor, se usa el tipo void para indicarlo.

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

- La función no tiene parámetros y no devuelve ningún valor explícito (void).

```
function saludar(): void {
  console.log('¡Hola!');
}
```

- La función toma dos parámetros de tipo number y devuelve un valor de tipo number.

```
function sumar(a: number, b: number): number {
  return a + b;
}
```

- La función obtenerNombre devuelve un valor de tipo string o undefined, utilizando la unión de tipos.

```
// Declaración de una función con tipo de retorno opcional
function obtenerNombre(): string | undefined {
  return 'Juan';
}
```

- La función crearSaludo tiene un parámetro opcional nombre, que puede ser string o undefined, y devuelve un saludo basado en ese nombre.

```
function crearSaludo(nombre?: string): string {
  if (nombre) {
    return `¡Hola, ${nombre}!`;
  } else {
    return '¡Hola!';
  }
}
```

Ejemplo:

```
function construirMensaje(nombre: string, edad?: number, genero: string = "Desconocido"): string {
  if (edad !== undefined) {
    return “Hola, ${nombre}. Tienes ${edad} años y eres ${genero}.”;
  } else {
    return “Hola, ${nombre}. No has proporcionado la edad, pero eres ${genero}.”;
  }
}
```

¿CÓMO ES EL PARÁMETRO NOMBRE? Obl + string
¿CÓMO ES EL PAARÁMETRO EDAD? Opc + number
¿CÓMO ES EL PARÁMETRO GÉNERO? Predeterminado y string

Ejemplo: Devuelve una cadena o un número dependiendo del resultado de la condición.

```
function calcularValor(valor: number): string | number {
  if (valor > 0) {
    return "Positivo";
  } else if (valor < 0) {
    return "Negativo";
  } else {
    return 0;
  }
}
```

5. Clases y Objetos:

a. Declaración de Clases:

Las clases son una característica fundamental de la programación orientada a objetos en TypeScript.

```
1. // Declaración de una clase
class Coche {
  // Propiedades
  marca: string;
  modelo: string;
  año: number;

2.  // Constructor
  constructor(marca: string, modelo: string, año: number) {
    this.marca = marca;
    this.modelo = modelo;
    this.año = año;
  }

3.  // Método
  obtenerInformacion(): string {
```

```
    return `${this.marca} ${this.modelo} (${this.año})`;
  }
}// Se cierra la clase
```

b. Propiedades y Métodos:

```
// Creación de un objeto a partir de la clase. INSTANCIA DE UNA CLASSE
const miCoche = new Coche("Toyota", "Corolla", 2022);

// Acceder a propiedades
console.log(`Marca: ${miCoche.marca}`);
console.log(`Modelo: ${miCoche.modelo}`);
console.log(`Año: ${miCoche.año}`);

// Llamar a método de la clase
const informacionCoche = miCoche.obtenerInformacion();
console.log(informacionCoche)
```

- Se crea un objeto `miCoche` a partir de la clase `Coche`.
- Se accede a las propiedades del objeto y se llama al método `obtenerInformacion`.

c. Herencia y Modificadores de Acceso: Se utiliza la palabra extends para indicar de dónde hereda

```
// Clase que hereda de Coche
class CocheDeportivo extends Coche {
  // Propiedad adicional
  velocidadMaxima: number;

  // Constructor propio
  constructor(marca: string, modelo: string, año: number, velocidadMaxima: number) {
    super(marca, modelo, año); // Llamada al constructor de la clase base
    this.velocidadMaxima = velocidadMaxima;
  }

  // Método propio
  acelerar(): void {
    console.log(`¡Acelerando a ${this.velocidadMaxima} km/h!`);
  }
}
```

Ejercicio 5. Gestión de alumnos. Implementa un sencillo sistema de gestión de alumnos con TypeScript, HTML y eventos de DOM. Cada alumno tendrá nombre, un edad y un titulación que está realizando. Se tiene que:

- **Crear una clase alumno con los atributos indicados, escogiendo el tipo que más se adecúe al caso.**
- **Implementar un método en la clase alumno llamado mostrarInformacion que devuelva la información del alumno en forma de cadena de texto.**
- **Crear un formulario HTML que permita introducir los datos del alumno.**

- **Implementar una función TypeScript que se ejecute al enviar el formulario y que cree una instancia de la clase `Alumno` con los datos ingresados, la agregue a una lista interna de alumnos y actualice la visualización de la lista de alumnos.**