**TypeScript**

**Tipo función.**

**Documentación:**

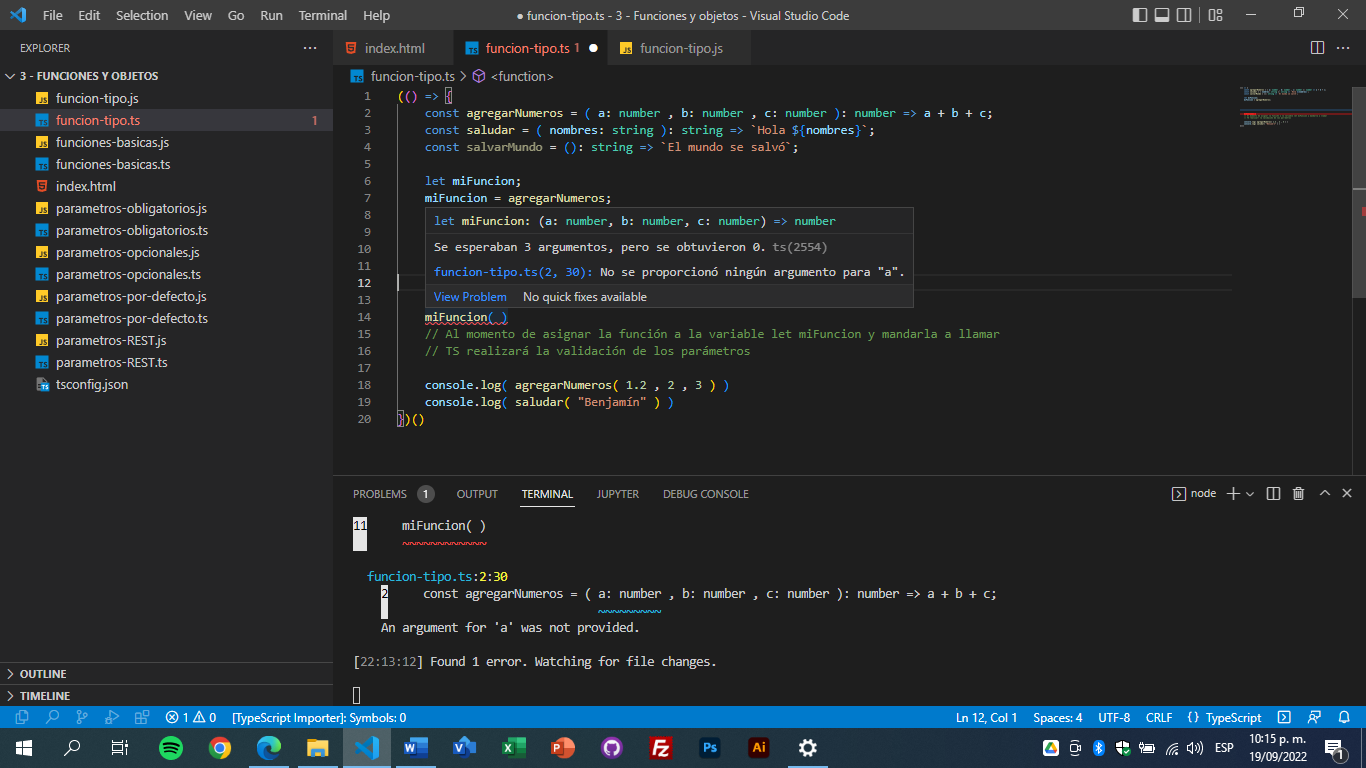
[**https://www.typescriptlang.org/docs/handbook/functions.html**](https://www.typescriptlang.org/docs/handbook/functions.html)

[**https://www.typescriptlang.org/docs/handbook/2/functions.html**](https://www.typescriptlang.org/docs/handbook/2/functions.html)

[**https://www.typescriptlang.org/docs/handbook/basic-types.html#object**](https://www.typescriptlang.org/docs/handbook/basic-types.html#object)

[**https://www.typescriptlang.org/docs/handbook/2/objects.html**](https://www.typescriptlang.org/docs/handbook/2/objects.html)

Al momento de utilizar variables para asignarles los tipos función y mandar a llamar las funciones con el nombre de la variable, TS realizará la validación de los parámetros infiriendo correctamente en los tipos de los parámetros aún después de su asignación, eso es lo que lo hace interesante.



Ejemplo del código:

(() => {

    const agregarNumeros = ( a: number , b: number , c: number ): number => a + b + c;

    const saludar = ( nombres: string ): string => `Hola ${nombres}`;

    const salvarMundo = (): string => `El mundo se salvó`;

    // Al momento de asignar la función a la variable let miFuncion y mandarla a llamar TS realizará la validación de los parámetros

    let miFuncion: any;

    miFuncion = agregarNumeros;

    console.log( miFuncion( 1 , 2 , 3 ) )

    miFuncion = saludar;

    console.log( miFuncion( "Benyi" ) )

    miFuncion = salvarMundo;

    console.log( miFuncion() )

})()

**Nota Importante.**

**Al utilizar los tipos función asignándolo en un variable no es conveniente utilizar el tipo any en la variable de asignación, sino especificar que vamos a utilizar un tipo función. Ejemplo:**

    const agregarNumeros = ( a: number , b: number ): number => a + b;

    const saludar = ( nombres: string ): string => `Hola ${nombres}`;

    const salvarMundo = (): string => `El mundo se salvó`;

let miFuncion: Function;

    miFuncion = agregarNumeros; // Asignación Ok

    console.log( miFuncion( 1 , 2 ) )

    miFuncion = saludar; // Asignación Ok

    console.log( miFuncion( "Benyi" ) )

    miFuncion = salvarMundo; // Asignación Ok

    console.log( miFuncion() )

    miFuncion = 10; // Asignación Error

**Para definir un tipo de retorno sobre un tipo de dato en específico que queremos retornar en nuestros tipos función y los tipos de datos de los parámetros que queremos enviar, como enviar numeros o cadenas y recibir un número o cadena, es decir, que cumpla la firma de la estructura de la función… Debemos borrar la palabra clave reservada “Function” y convertir la variable de asignación en una función, especificando los tipos… Ejemplo:**

const agregarNumeros = ( a: number , b: number ): number => a + b;

    const saludar = ( nombres: string ): string => `Hola ${nombres}`;

    const salvarMundo = (): string => `El mundo se salvó`;

// Especificación de la firma de los tipos de datos a enviar como parámetros y recibir como retorno en el tipo función...

    let miFuncion2: (x: number, y: number) => number;

    miFuncion2 = agregarNumeros; // Asignación Ok

    console.log( miFuncion( 1 , 2 ) )

    miFuncion2 = saludar; // Asignación Error

    console.log( miFuncion( "Benyi" ) )

    miFuncion2 = salvarMundo; // Asignación Error

    console.log( miFuncion() )

    miFuncion2 = 10; // Asignación Error

**Se pueden recibir cualquier tipo de retorno en las firmas del tipo función…**

    let miFuncion2: (x: number, y: number) => number;

    let miFuncion2: (y: string) => string;

    let miFuncion2: (y: number) => void;