# CÓMO USAR PROMESAS DESDE TYPESCRIPT SIN ERRORES:

Si queréis usar promesas desde Visual Studio Code y con Typescript, probablemente os de un error de compilación, aunque genera el fichero .js y lo ejecuta bien de todas formas.

Pero si queréis que compile sin errores. Podéis hacer lo siguiente.

DESDE TYPESCRIPT 2.0 ES CUANDO HAY PROBLEMAS.

Si da el error:

```
prueba.ts:2:16 - error TS2585: 'Promise' only refers to a type, but is
being used as a value here. Do you need to change your target library?
Try changing the `lib` compiler option to es2015 or later.
```

Es porque TypeScript está configurado para compilar a EcmaScript 5 y las promesas no salieron hasta el ES6 (ó ES2015; *es lo mismo). Mirad vuestro fichero "tsconfig.json":*

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs"
  }
}
```

## *POSIBLES SOLUCIONES A ESTE ERROR:*

### 1. Modificar el fichero "tsconfig.json" para añadir las librerías necesarias a incluir en la compilación para que entienda las promesas.

En concreto, se necesita la librería ES2015 y DOM como mínimo. Debéis añadir lo siguiente en "compilerOptions":

```
{
  "compilerOptions": {
    /* Visit https://aka.ms/tsconfig.json to read more about this file */

    /* Basic Options */
```

```
    // "incremental": true,                   /* Enable incremental compilatio
n */
    "target": "es5",                       /* Specify ECMAScript target version:
 'ES3' (default), 'ES5', 'ES2015', 'ES2016', 'ES2017', 'ES2018', 'ES2019', 'ES
2020', or 'ESNEXT'. */
    "lib":["es2015","dom"],
    "module": "commonjs",                  /* Specify module code generatio
n: 'none', 'commonjs', 'amd', 'system', 'umd', 'es2015', 'es2020', or 'ESNext'
. */
    // "lib": [],
```

Y ahora para transpilar (*compilar a JS), se hace sin indicar el nombre del archivo para que use ese fichero tsconfig.json y no el que trae por defecto*.

Podéis transpilar de 2 formas:

a) *Escribiendo:*
   > *tsc* → *Esto compila todos los archivos .ts que haya en esa carpeta usando el tsconfig.json que haya en esa carpeta*

b) *Escribiendo:*
   > *tsc --project tsconfig.json* → *Esto compila todos los archivos .ts que haya en esa carpeta usando el tsconfig.json indicado aquí*

## 2. NO MODIFICAR ningún fichero ni nada y a la hora de transpilar indicar qué librería/s queremos usar.

Simplemente escribimos en la terminal de Visual Studio Code:

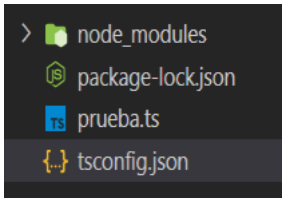
> tsc –lib "es2015,dom" → eso le indica que compile todo lo que hay en esa carpeta donde estamos usando las librerías “es2015” y la “dom”

## 3. NO MODIFICAR ningún fichero, pero instalar @types/node que son tipos de datos de ES6 y de node (incluyen las promesas, entre otros).

Para instalarlos, escribimos en la consola:

> npm i @types/node

Esto nos crea una carpeta “node_modules” y un “package-lock.json” en nuestra carpeta de proyecto con todos esos tipos incluidos:




Ahora simplemente compilamos de forma normal, escribiendo:

> tsc  prueba.ts

## *CURIOSIDAD: CÓMO INDICAR QUE UNA FUNCIÓN ES DE TIPO PROMESA EN TYPESCRIPT Y QUÉ DEVUELVE LA PROMESA:*

Para el examen, no se os va a exigir que declaréis los tipos en las promesas, pero si alguien tiene curiosidad, se haría así:

```
function esMayorEdad(edad:number):Promise<boolean>  {
    return new Promise <boolean>((resolve, reject) => {
        if (edad >= 18)
             resolve(true);
        else
            reject(false);
    })
}

// PRUEBA DE LLAMADA-usando fórmula 1 de escribir el then y catch
let edad1:number=19;
esMayorEdad(edad1)
    .then(resultado => console.log(edad1 + " SÍ ES MAYOR EDAD"))
    .catch(resultado => console.log(edad1+" NO ES MAYOR EDAD"));
```

Como veis en la imagen superior resaltado, se indica que se devuelve una promesa, que a su vez devuelve un booleano. Además, al hacer new Promise también se indica qué tipo de dato devuelve esa promesa que estamos creando con los símbolos <tipo>.