


✓ 100 XP 

# Compilación de un archivo en TypeScript

5 minutos

Como ya hemos visto, TypeScript es un supraconjunto estricto de [ECMAScript 2015](#) (ECMAScript 6 o ES6). Todo el código JavaScript es también código TypeScript, por lo que un programa escrito en TypeScript puede consumir JavaScript sin problemas. De hecho, puede convertir un archivo JavaScript en TypeScript simplemente cambiando el nombre de la extensión de `.js` a `.ts`.

Aun así, no todo el código TypeScript es JavaScript. TypeScript agrega sintaxis nueva a JavaScript, lo que facilita la lectura del código JavaScript e implementa algunas características, como los tipos estáticos. Aunque el código TypeScript hace que el desarrollo sea más sencillo y menos propenso a errores, los exploradores y la mayoría de los demás entornos de ejecución no admiten TypeScript de forma nativa. Por esta razón, TypeScript requiere un paso de compilación ([transpilador](#) ) para transformarlo en JavaScript para que la aplicación funcione.

Para transformar código TypeScript en JavaScript, use el compilador de TypeScript o un transpilador compatible con TypeScript, como [Babel](#) , [swc](#) o [Sucrase](#) . Este proceso quita el código específico de TypeScript (por ejemplo, las interfaces y las declaraciones de tipos). Además, genera un archivo JavaScript limpio que puede ejecutar en las páginas web y que es compatible con los exploradores.

## Compilación de un archivo en TypeScript

Para ejecutar el compilador de TypeScript en el símbolo del sistema, use el comando `tsc` . Cuando se ejecuta `tsc` sin parámetros, se compilan todos los archivos `.ts` en la carpeta actual y se genera un archivo `.js` para cada uno.

También puede compilar un archivo específico. Por ejemplo, para compilar un archivo TypeScript denominado `utility_functions.ts`, escriba `tsc utility_functions.ts` .

### Nota

Es opcional escribir la extensión de archivo `.ts`.

Si no hay ningún error del compilador, el comando `tsc` genera un archivo JavaScript denominado *utility\_functions.js*.

Si el compilador encuentra errores en el código, los muestra en la ventana de comandos. Corrija los errores del archivo TypeScript y vuelva a ejecutar el comando `tsc`.

## Opciones del compilador

Las opciones del compilador le permiten controlar cómo se genera el código JavaScript a partir del código TypeScript de origen. Puede establecer las opciones en el símbolo del sistema, como haría en el caso de muchas interfaces de la línea de comandos, o en un archivo JSON denominado *tsconfig.json*.

Hay disponibles numerosas opciones del compilador. Encontrará una lista completa de opciones en la [documentación de las interfaces de la línea de comandos de tsc](#). Estas son algunas de las opciones más comunes:

- `noImplicitAny`
- `noEmitOnError`
- `target`
- Opciones de directorio

Para controlar la compilación, puede usar las opciones del compilador con el comando `tsc`, incluidas las siguientes:

- La opción `--noImplicitAny` indica al compilador que genere errores en expresiones y declaraciones con un tipo `any` implícito. Por ejemplo:

```
tsc utility_functions.ts --noImplicitAny
```

- La opción `--target` especifica la versión de destino de ECMAScript para el archivo JavaScript. En este ejemplo se compila un archivo JavaScript compatible con ECMAScript 6:

```
tsc utility_functions.ts --target "ES2015"
```

Descubrirá otras opciones del compilador en módulos posteriores.

## Siguiente unidad: Ejercicio: Configuración de un proyecto de TypeScript

[Continuar >](#)

---

¿Qué tal lo estamos haciendo? ☆ ☆ ☆ ☆ ☆