



# MANUAL NODE.JS

## Instalación de TypeScript

### Descripción breve

En este manual veremos como instalar node.js en local y su uso. Instalaremos el paquete de TypeScript y haremos uso de él.

Emilio Sánchez

SGDB – 2º ASIR

---

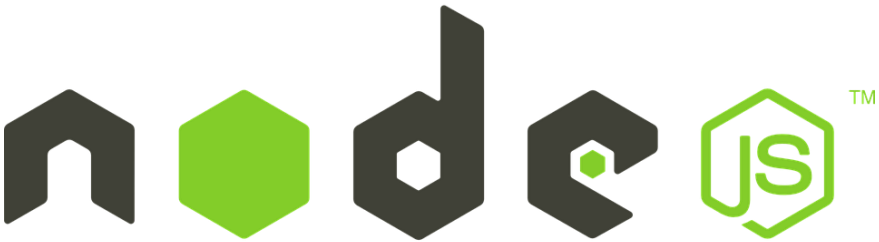
# ÍNDICE

1. Introducción
2. Instalación node.js y npm
3. Instalación TypeScript
  - a. Global
  - b. DevDependency
  - c. Desinstalar
4. TypeScript
  - a. Ejemplos
  - b. Compilación
  - c. npm y tsc

---

# INTRODUCCIÓN

Node.js es un entorno en tiempo de ejecución multiplataforma, de código abierto, para la capa del servidor basado en el lenguaje de programación JavaScript.



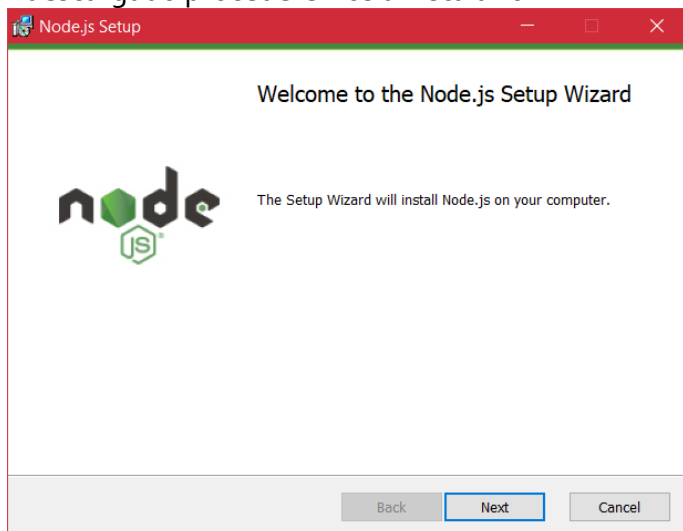
TypeScript es un lenguaje de programación libre y de código abierto desarrollado y mantenido por Microsoft. Es un superconjunto de JavaScript, que esencialmente añade tipos estáticos y objetos basados en clases.

# INSTALACIÓN NODE.JS Y NPM

Iremos a la [página oficial](#) de **node.js** para instalar la última versión estable.

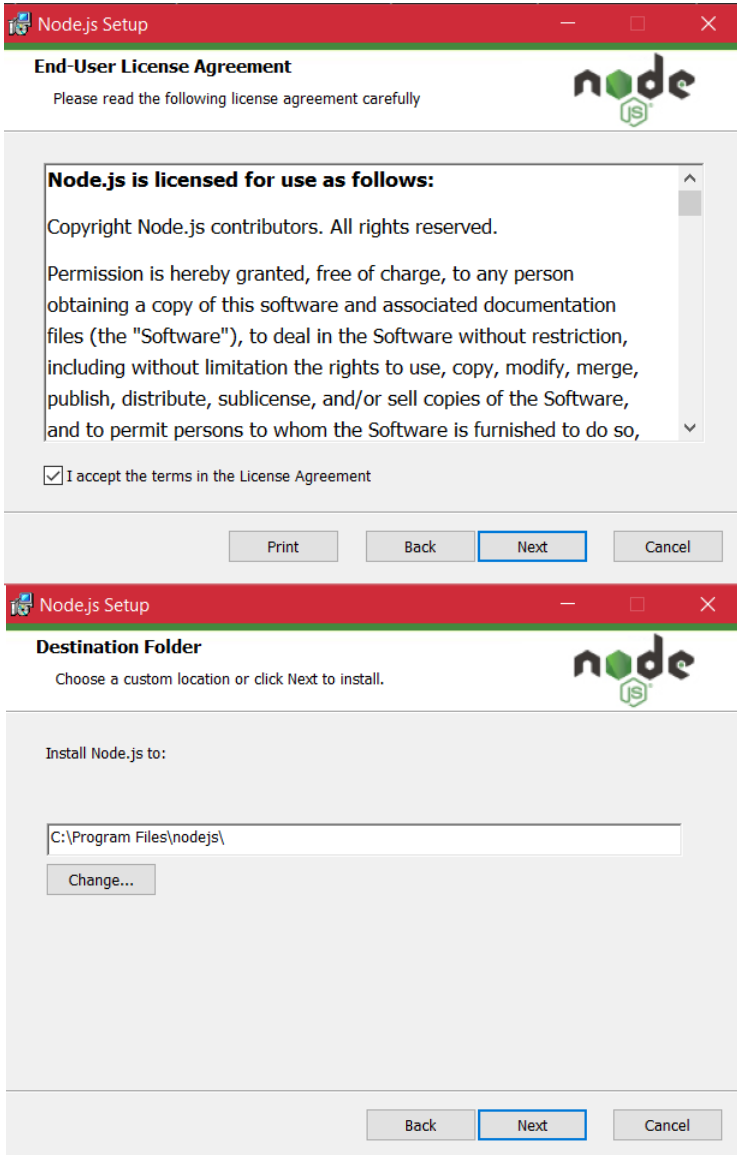


Una vez descargado procederemos a instalarlo.

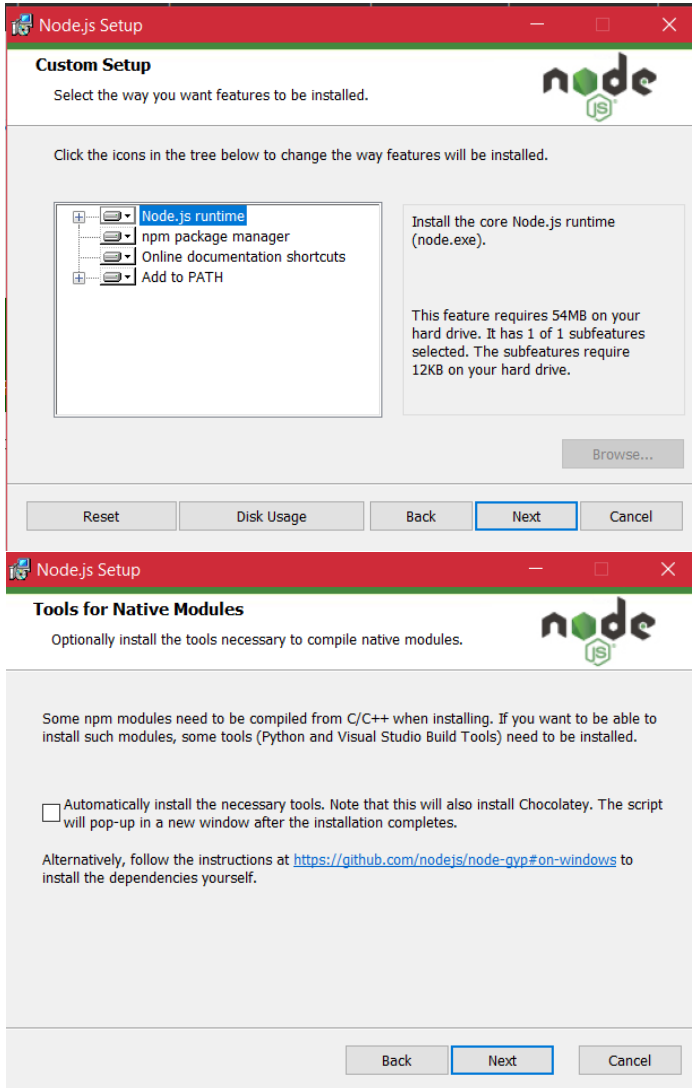


# INSTALACIÓN NODE.JS Y NPM

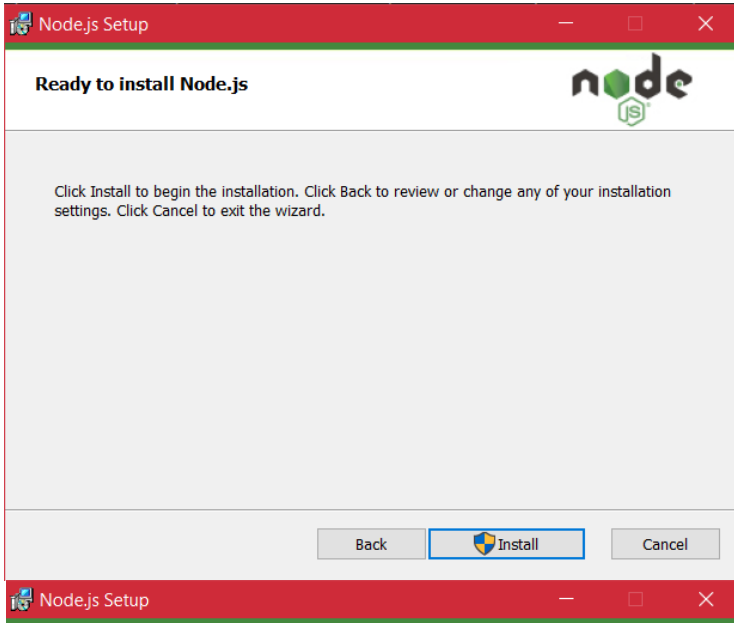
Clicaremos en **Next** en todo momento.



# INSTALACIÓN NODE.JS Y NPM



# INSTALACIÓN NODE.JS Y NPM

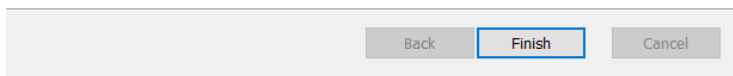


Completed the Node.js Setup Wizard



Click the Finish button to exit the Setup Wizard.

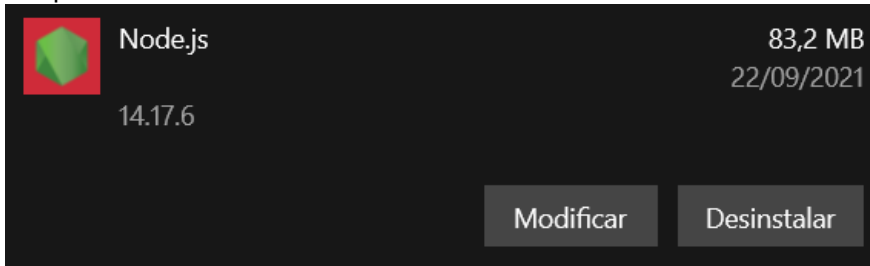
Node.js has been successfully installed.



---

# INSTALACIÓN NODE.JS Y NPM

Podemos comprobar la instalación desde la lista de programas simplemente.



O desde la PowerShell.

```
PS C:\Users\emili> node
Welcome to Node.js v14.17.6.
Type ".help" for more information.
>
```



---

# INSTALACIÓN NODE.JS Y NPM

El paquete **npm** (Node Package Manager) nos viene instalado junto a node.js. Este nos servirá para poder instalar dependencias de node.js.

```
PS C:\Users\emili> npm

Usage: npm <command>

where <command> is one of:
  access, adduser, audit, bin, bugs, c, cache, ci, cit,
  clean-install, clean-install-test, completion, config,
  create, ddp, dedupe, deprecate, dist-tag, docs, doctor,
  edit, explore, fund, get, help, help-search, hook, i, init,
  install, install-ci-test, install-test, it, link, list, ln,
  login, logout, ls, org, outdated, owner, pack, ping, prefix,
  profile, prune, publish, rb, rebuild, repo, restart, root,
  run, run-script, s, se, search, set, shrinkwrap, star,
  stars, start, stop, t, team, test, token, tst, un,
  uninstall, unpublish, unstar, up, update, v, version, view,
  whoami

npm <command> -h  quick help on <command>
npm -l           display full usage info
npm help <term>  search for help on <term>
npm help npm     involved overview

Specify configs in the ini-formatted file:
  C:\Users\emili\.npmrc
or on the command line via: npm <command> --key value
Config info can be viewed via: npm help config

npm@6.14.15 C:\Program Files\nodejs\node_modules\npm
```

---

# INSTALACIÓN DE TYPESCRIPT

## GLOBAL

Podemos instalar TypeScript en global, lo que significa que estará en nuestro dispositivo entero.

El comando que usaremos será: **npm install -g typescript**

```
PS C:\Users\emili> npm install -g typescript
C:\Users\emili\AppData\Roaming\npm\tsserver -> C:\Users\emili\AppData\Roaming\npm\node_modules\typescript\bin\tsserver
C:\Users\emili\AppData\Roaming\npm\tsc -> C:\Users\emili\AppData\Roaming\npm\node_modules\typescript\bin\tsc
+ typescript@4.4.3
added 1 package from 1 contributor in 2.493s
```

## DEVDEPENDENCY

Con DevDependency instalaremos TypeScript en nuestro proyecto, eligiendo nosotros las carpetas que deseemos.

Para ello iremos a la carpeta donde tengamos nuestro proyecto y abriremos la terminal. Escribiremos la línea: **npm install typescript**

**--save-dev**

```
PS C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer trimestre\24-09-2021 Proyecto1> npm install typescript --save-dev
npm WARN saveError ENOENT: no such file or directory, open 'C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer trimestre\24-09-2021 Proyecto1\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer trimestre\24-09-2021 Proyecto1\package.json'
npm WARN 24-09-2021 Proyecto1 No description
npm WARN 24-09-2021 Proyecto1 No repository field.
npm WARN 24-09-2021 Proyecto1 No README data
npm WARN 24-09-2021 Proyecto1 No license field.

+ typescript@4.4.3
added 1 package from 1 contributor and audited 1 package in 2.927s
found 0 vulnerabilities
```

Esto nos creará una nueva carpeta:

```
> node_modules
```

---

# INSTALACIÓN DE TYPESCRIPT

## DESINSTALAR

Si queremos desinstalar el paquete global usaremos este comando.

```
PS C:\Users\emili> npm uninstall -g typescript
removed 1 package in 0.156s
```

Si queremos desinstalar el paquete en DevDependency usaremos el mismo sin `-g`.

Para desinstalar en local usaremos:

```
PS C:\miscosas\01-Estudios\01-SegAsir\SGBO\Primer_trimestre\24-09-2021_Proyecto> npm uninstall typescript
npm WARN saveError ENOENT: no such file or directory, open 'C:\miscosas\01-Estudios\01-SegAsir\SGBO\Primer_trimestre\24-09-2021_Proyecto\package.json'
npm WARN enoent ENOENT: no such file or directory, open 'C:\miscosas\01-Estudios\01-SegAsir\SGBO\Primer_trimestre\24-09-2021_Proyecto\package.json'
npm WARN 24-09-2021_Proyecto No description.
npm WARN 24-09-2021_Proyecto No repository field.
npm WARN 24-09-2021_Proyecto No README data
npm WARN 24-09-2021_Proyecto No license field.

removed 1 package in 0.519s
found 0 vulnerabilities
```

# Typescript

## EJEMPLOS

Usaremos estos archivos como ejemplo:

```
TS ejercicio01.ts X JS ejercicio01.js
SGBD > Primer_trimestre > 22-09-2021_Practica2
1 console.log("Hola mundo TS")
2
```

```
TS ejercicio01.ts JS ejercicio01.js X
SGBD > Primer_trimestre > 22-09-2021_Practica2
1 console.log("Hola mundo JS")
2
```

Realmente el archivo `.js` no se trata de JavaScript. Con node ya podremos comprobar que funcionen estos archivos.

```
PS C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer_trimestre\22-09-2021_Practica2\src> node ejercicio01.ts
Hola mundo TS
PS C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer_trimestre\22-09-2021_Practica2\src> node ejercicio01.js
Hola mundo JS
```

## COMPILACIÓN

Al usar TypeScript muchos navegadores no son capaces de leerlo, por lo que es necesario pasarlo a un lenguaje más común, como JavaScript.

Para ello compilaremos el código de TypeScript. Para compilar usaremos el comando `tsc`. Al intentarlo la primera vez nos devolverá un error de permisos.

```
PS C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer_trimestre\22-09-2021_Practica2\src> tsc ejercicio02.ts
tsc : No se puede cargar el archivo C:\Users\emili\AppData\Roaming\npm\tsc.ps1 porque la ejecución de scripts está
deshabilitada en este sistema. Para obtener más información, consulta el tema about_Execution_Policies en
https://go.microsoft.com/fwlink/?LinkID=135170.
En línea: 1 Carácter: 1
+ tsc ejercicio02.ts
+ ~~~~
+ CategoryInfo          : SecurityError: (:) [], PSSecurityException
+ FullyQualifiedErrorId : UnauthorizedAccess
```

# Typescript

Navegaremos a esta [página](#) de documentación de Microsoft. En ella nos hablarán del comando **Set-ExecutionPolicy**. Esto nos permite asignar permisos de ejecución de scripts.

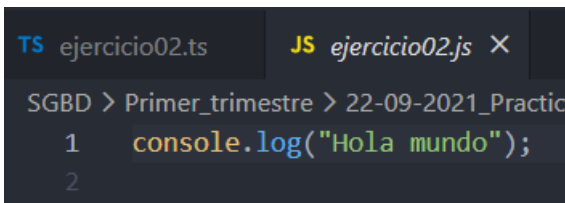
Usaremos el de **unrestricted** para tener mayor libertad.

```
PS C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer_trimestre\22-09-2021_Practica2\src> Set-ExecutionPolicy unrestricted

Cambio de directiva de ejecución
La directiva de ejecución te ayuda a protegerte de scripts en los que no confías. Si cambias dicha directiva, podrías exponerte a los riesgos de seguridad descritos en el tema de la Ayuda about_Execution_Policies en https://go.microsoft.com/fwlink/?LinkID=135170. ¿Quieres cambiar la directiva de ejecución?
[S] Sí [O] Sí a todo [N] No [T] No a todo [U] Suspender [?] Ayuda (el valor predeterminado es "N"): S
```

Nos preguntará por la directiva, en este caso elegiremos simplemente **Sí**, pero si da error se podría usar **Sí a todo**.

Compilaremos ahora nuevamente (usamos un archivo distinto pero con el mismo código). Como podemos ver, el código devuelto por el archivo compilado en JavaScript es distinto al que nosotros hicimos.



```
TS ejercicio02.ts JS ejercicio02.js X
SGBD > Primer_trimestre > 22-09-2021_Practica2
1 console.log("Hola mundo");
2
```

Existe otra opción que es la de **tsc -w**, en la que estará continuamente buscando cambios en los archivos para compilarlos al momento, en vez de hacerlo manualmente.

---

# Typescript

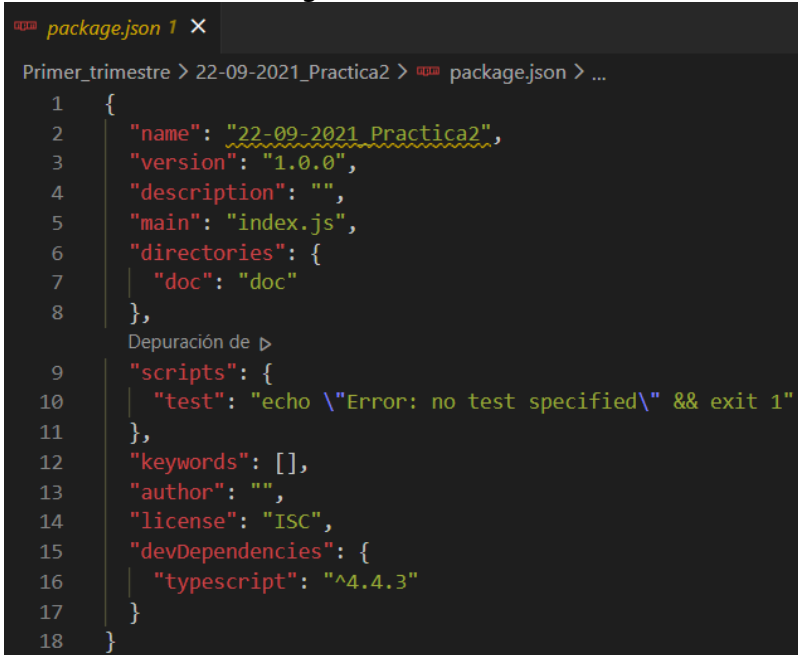
## NPM Y TSC

Existen ciertos archivos de gestión en node.js.

Empezaremos por **package.json**. Para ello nos dirigiremos a la terminal del proyecto e iniciaremos npm.

Usaremos el comando: **npm init -y** (*usaremos el -y para que no nos pregunte*)

Al hacerlo nos creará el siguiente archivo:



```
1 {
2   "name": "22-09-2021_Practica2",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "directories": {
7     "doc": "doc"
8   },
9   "scripts": {
10    "test": "echo \"Error: no test specified\" && exit 1"
11  },
12  "keywords": [],
13  "author": "",
14  "license": "ISC",
15  "devDependencies": {
16    "typescript": "^4.4.3"
17  }
18 }
```

Desde este archivo podremos controlar todos los paquetes de npm, versiones y demás información.

# Typescript

Si usamos **tsc --init**, con este comando crearemos el archivo **tsconfig.json**.

```
PS C:\miscosas\01-Estudios\01-SegAsir\SGBD\Primer trimestre\22-09-2021_Practica2> tsc --init
message TS6071: Successfully created a tsconfig.json file.
```

**tsconfig.json**

```
1 {
2   "compilerOptions": {
3     /* Visit https://aka.ms/tsconfig.json to read more about this file */
4
5     /* Projects */
6     // "incremental": true,           /* Enable incremental compilation */
7     // "composite": true,            /* Enable constraints that allow a TypeScript project to be used with project re
8     // "tsBuildInfoFile": ".",        /* Specify the folder for .tsbuildinfo incremental compilation files. */
9     // "disableSourceOfProjectReferenceRedirect": true, /* Disable preferring source files instead of declaration files when referencing
10    // "disableSolutionSearching": true, /* Opt a project out of multi-project reference checking when editing. */
11    // "disableReferencedProjectLoad": true, /* Reduce the number of projects loaded automatically by TypeScript. */
12
13    /* Language and Environment */
14    "target": "es5",                 /* Set the JavaScript language version for emitted JavaScript and include compat
15    // "lib": [],                     /* Specify a set of bundled library declaration files that describe the target r
16    // "jsx": "preserve",             /* Specify what JSX code is generated. */
17    // "experimentalDecorators": true, /* Enable experimental support for TC39 stage 2 draft decorators. */
18    // "emitDecoratorMetadata": true, /* Emit design-type metadata for decorated declarations in source files. */
19    // "jsxFactory": "",              /* Specify the JSX factory function used when targeting React JSX emit, e.g. 'Re
20    // "jsxFragmentFactory": "",      /* Specify the JSX Fragment reference used for fragments when targeting React JS
21    // "jsxImportSource": "",         /* Specify module specifier used to import the JSX factory functions when using
22    // "reactNamespace": "",          /* Specify the object invoked for "createElement". This only applies when target
23    // "noLib": true,                 /* Disable including any library files, including the default lib.d.ts. */
```

Para tener una mayor compatibilidad con los distintos navegadores podemos cambiar de target es5 a **es6**.

```
"target": "es6",
```

Buscaremos luego la línea de outDir.

```
// "outDir": ".",
```

Borraremos las barras que comentan la línea y elegiremos una carpeta (generalmente **dist**, distribution).

Esta carpeta se creará cuando compilemos el código, llevando hacia ella todos los ficheros JavaScript compilados de TypeScript. Si no se marca, se compilarán en la misma carpeta que el TypeScript.