# Introducció a Typescript

Typescript

Duració: 2h

# Introducció

TypeScript és un superset de JavaScript creat per Microsoft, que afegeix tipatge estàtic i altres característiques avançades al llenguatge.

En altres paraules, TypeScript és JavaScript amb tipus de dades i altres millores per fer el desenvolupament més segur i fàcil de mantenir.

El codi escrit en TypeScript es compila a JavaScript, de manera que és compatible amb qualsevol navegador o entorn que suporti JavaScript.

# Característiques de TypeScript

**Tipatge estàtic**: TypeScript permet declarar el tipus de les variables (com a string, number, boolean, etc.).

**Classes i interfaces**: TypeScript ofereix suport per a la programació orientada a objectes amb classes, interfaces i herència.

**Compatibilitat amb JavaScript**: Tot el codi JavaScript és vàlid en TypeScript.

**Autocompletat i intel·ligència de codi**: Amb el tipatge explícit, els editors de codi (com Visual Studio Code) poden oferir una millor autocompletació i suggeriments

**Mòduls i espais de noms**: TypeScript suporta la creació de mòduls i l'ús d'espais de noms.

**Compilació a JavaScript**: El codi TypeScript es compila a JavaScript abans de ser executat. La compilació permet detectar errors de tipatge i altres problemes abans de l'execució.

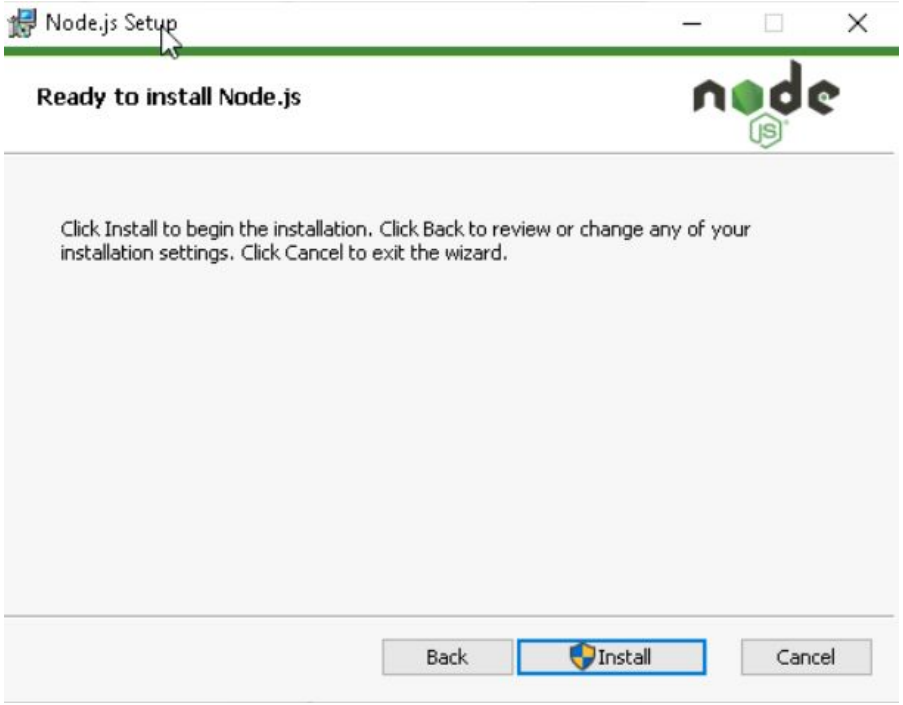
# Compilador

El compilador de TypeScript anomenat **`tsc`** és una eina que converteix el codi escrit en TypeScript en codi JavaScript.

Per poder fer-ho servir s'ha d'instal·lar Node js, el qual instal·larà el gestor de paquets npm.

Un cop instal·lat podem instal·lar el compilador amb la següent comanda:

```
npm install -g typescript
```



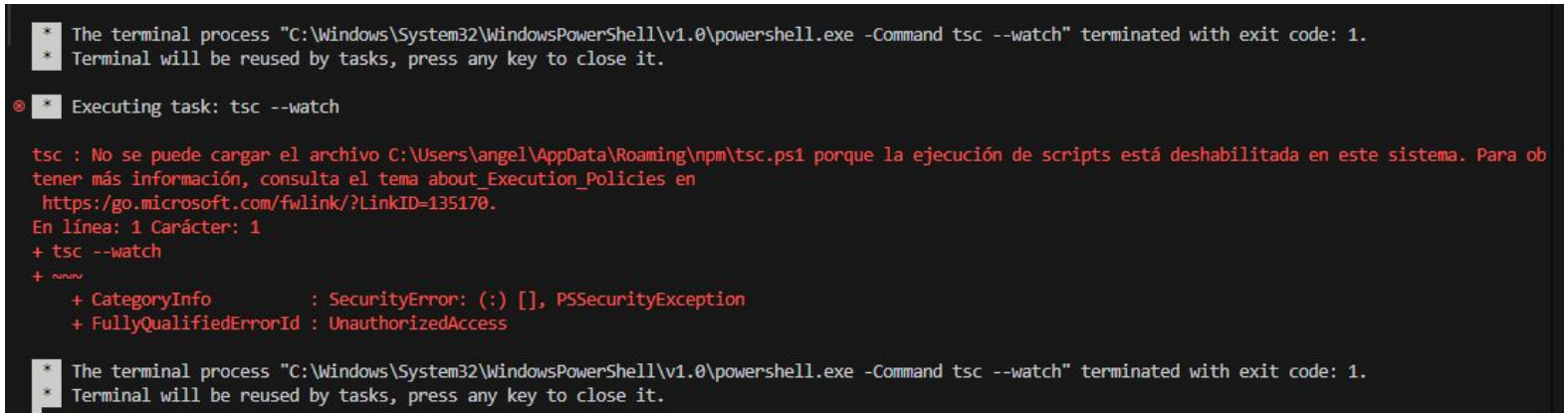
# Dehabilitar politiques de Powersell

Si fas servir Powershell com terminal per defecte s'ha de desactivar la politica d'execució d'escripts per poder instal·lar node i executar tsc.

Executa la següent comanda com administrador:

```
Set-ExecutionPolicy unrestricted //Desactiva la restricció
```

```
Set-ExecutionPolicy restricted //Activa la restricció
```



# Inici del projecte

Typescript compila el seu codi en codi javascipt per tal de que el navegador executi un codi que aquest pugui entendre.

La comanda tsc necessita un fitxer de configuració  per crear-ho s'ha d'executar la següent comanda:

```
tsc –init
```

tsconfig.json

```
{
 "compilerOptions": {
   "target": "es2020",
   "module": "commonjs",
   "esModuleInterop": true,
   "forceConsistentCasingInFileNames": true,
   "strict": true,
   "skipLibCheck": true
 }
}
```

# Compilació del codi

Al fer un fitxer amb extensió .ts el compilador compilarà el codi del fitxer i el guardarà en un altre fitxer .js amb el codi javascript per poder exectuar des del navegador. Per executar la compilació s'ha d'executar la següent comanda:

```
tsc
```

app.ts

```
let value : string = "test"

function print (...params: string[]) : void{
   console.log(...params)
}

print(value)
```

També es pot fer servir -w per tal de deixar el compilador executan-se en busca de canvis als documents i compilar-los automàticament:

```
tsc -w
```

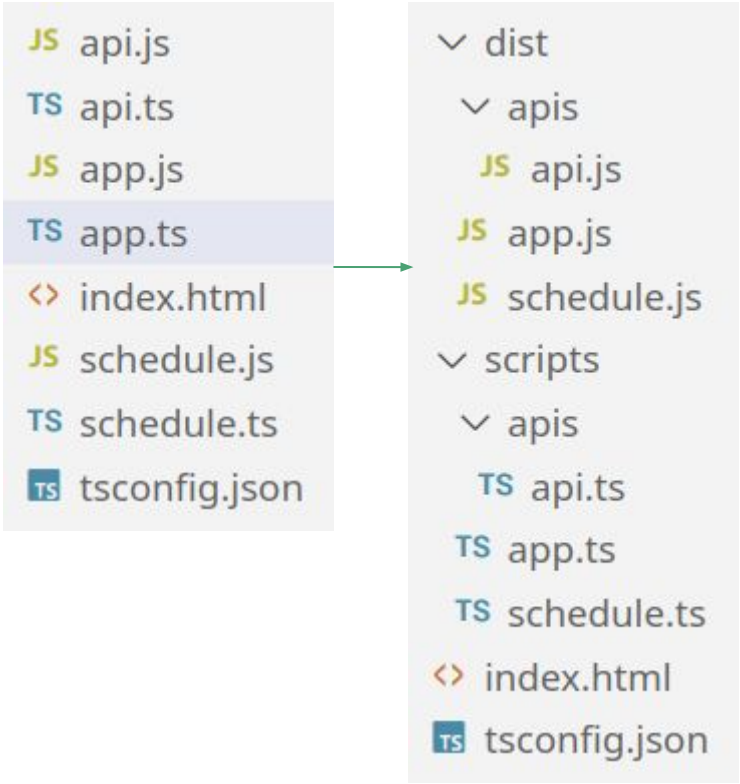
app.js

```
let value = "test";
function print(...params) {
   console.log(...params);
}
```

# Organització

Quan el projecte va creixent la gestió dels fitxers es va fent més complex. Podem modificar el fitxer de configuració per tal de guardar tots els fitxers js en una carpeta a part. per fer això s'ha d'afegir la següent línia de configuració al fitxer tsconfig.json:

```
"outDir": "./dist"
```



# Depuració en VSCode

Per poder debugar des de vscode hem de afegir algunes configuracions:

1. Mapejat del fitxers.
2. Configuració del launcher per obrir el navegador en mode de depuració.
3. Configuració de la tasca de compilació.

# Depuració: Mapejat del fitxers.

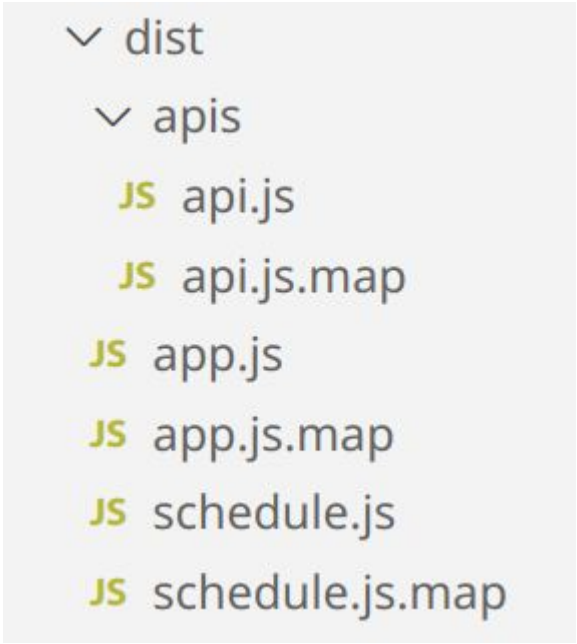
Modificar el fitxer tsconfig.json per afegir el mapejat del fitxers.

```
"sourceMap": true
```

mapejat un cop configurat si executem la comanda:

```
tsc
```

Hem de veure els fitxers del mapejat.



# Depuració: Configuració del launcher

Al fitxer de .vscode/launch.json

Afegir el següent:

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "chrome",
            "request": "launch",
            "name": "Launch Chrome",
            "file": "${workspaceFolder}/index.html" ,
        }
    ]
}
```

# Depuració: Configuració de la tasca de compilació

Crear el fitxer .vscode/tasks.json

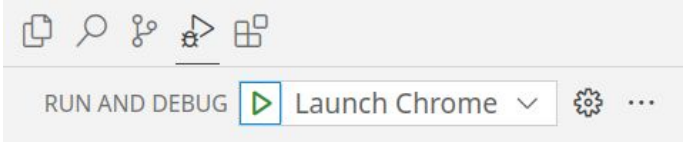
```
{
   "version": "2.0.0",
   "tasks": [
     {
       "label": "tsc: compile",
       "type": "shell",
       "command": "tsc",
       "args": [
         "--watch"
       ],
       "group": {
         "kind": "build",
         "isDefault": true
       },
       "problemMatcher": ["$tsc-watch"],
       "isBackground": true,
       "runOptions": {
         "runOn": "folderOpen"
       }
     }
   ]
 }
```

Modifiquem el Fitxer .vscode/launch.json

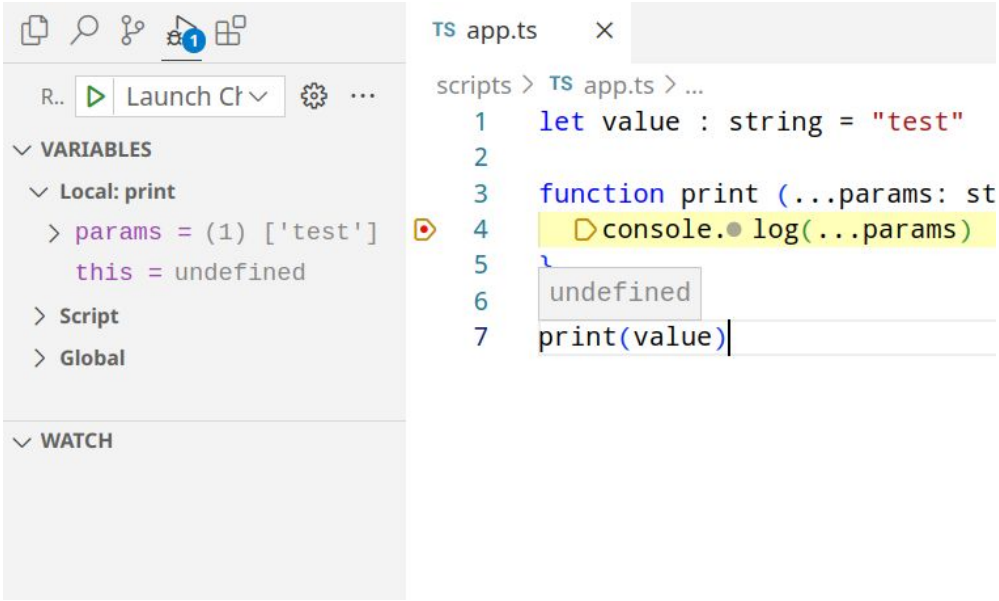
```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "chrome",
            "request": "launch",
            "name": "Launch Chrome",
            "file":
"${workspaceFolder}/index.html",
            "preLaunchTask": "tsc: compile"
        }
    ]
}
```

# Depuració

Un cop tenim tot preparat per poder debugar ens ha d'apareixer a l'apartat de depuració la nova configuració.




Fem clic al botó d'iniciar debugacció i podem posar directament punts de parada als fitxers de typescript.



# ES6 vs commonjs

Per raons històriques i de compatibilitat amb node.js és defineix que la gestió d'importació de mòduls és per defecte **`commonjs`**, però per aprofitar les funcionalitats més noves farem servir la importació de mòduls **ES6**.

Modifiquem el atribut module del fitxer de configuració de typescript **`tsconfig.json`** amb el valor **`ES6`**.

tsconfig.json

```
{
 "compilerOptions": {
   "target": "es2016",
   "module": "ES6",
   "esModuleInterop": true,
   "forceConsistentCasingInFileNames": true,
   "strict": true,
   "skipLibCheck": true
 }
}
```

# Configuració per poder fer servir importació de mòduls amb Typescript

# Servidor local (I)

La importació de mòduls de typescript per raons de seguretat només funciona sota el mateix domini. Això vol dir que per poder fer importacions de mòduls hem de servir la nostra web fent servir un servidor web.

Per aquest motiu instal·larem el següent paquet:

```
npm install -g live-server
```

Un cop instal·lat modifcarem el fitxers task.json per afegir l'execució del live-server.

```
{
   "version": "2.0.0",
   "tasks": [
     { //tsc: compiler },
     {
       "label": "tsc watch + Live Server" ,
       "dependsOn": ["tsc: compile", "Start Live Server"],
       "dependsOrder": "sequence"
     },
     {
       "label": "Start Live Server",
       "type": "shell",
       "command": "npx",
       "args": ["live-server", "--port=5500", "--open=index.html"],
       "isBackground": true,
       "problemMatcher": [
         {
           "pattern": [
             {
               "regexp": ".",
               "file": 1,
               "location": 2,
               "message": 3
             }
           ],
           "background": {
             "activeOnStart": true,
             "beginsPattern": ".",
             "endsPattern": "."
           }
         }
       ]
     }
   ]
 }
```

# Servidor local (II)

Al fitxer launch.json modificarem el fitxer **`launch.json`** i modificarem el atribut **`preLaunchTask`** per executar la tasca composta que hem creat el punt anterior.

D'altra banda ja no s'executarà des de el fitxer si no des de un servidor la url del nostre servidor serà **`http://localhost:5500`**

Per fer això s'ha de treure el atribut **`file`** i afegim els atributs **`url`** i **`webRoot`**.

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "type": "chrome",
            "request": "launch",
            "name": "Launch Chrome",
            "url":"http://localhost:5500",
            "webRoot": "${workspaceFolder}",
            "preLaunchTask": "tsc watch + Live Server"
        }
    ]
}
```

# Importació de mòduls amb tsc

Tsc no modifica l'extensió d'un mòdul exportat la sintaxi que ens autogenera vscode quan importa un mòdul és:

```
import add from "./foo"
```

Però al executar al servidor ens donà un error que no troba el fitxer foo i és normal, ja que tsc l'ha convertit en foo.js. S'ha d'indicar de la següent forma:

```
import add from "./foo.js"
```

# Transpiladors

Existeixen diferents transpiladors:

- tsc: Transpilador oficial de TypeScript.
- Babel: Transpilador de JavaScript que també pot treballar amb TypeScript.
- Rollup: Empaquetador de mòduls JavaScript modern.
- ESM natiu: Empaquetador que fa servir Vite

Per poder solucionar l’inconvenient anterior es pot fer servir Babel o Vite