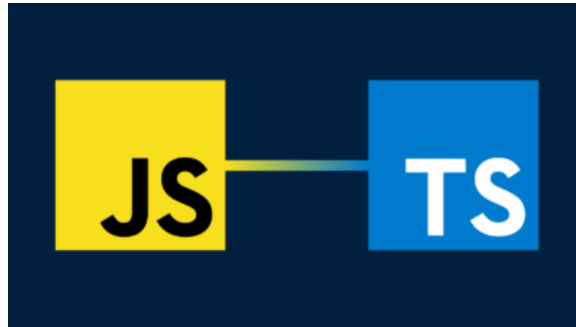


# Introduction à TypeScript



Prérequis - JavaScript - NodeJS

Objectif : installer TypeScript

Durée : 3h30



Fichiers source sur Git :

Exemple 1 - Conversion de formulaire JavaScript en TypeScript

<https://github.com/citywizz/js-to-ts-form>

Exemple 2 - Structure d'un projet TypeScript

<https://github.com/citywizz/ts-default-project>

## 1. Qu'est-ce que TypeScript ?

TypeScript est un langage de programmation qui complète JavaScript. Il a été co-crée par Anders Hejlsberg, principal inventeur de C# en 2012.

Les nouvelles fonctionnalités ajoutées à JavaScript :

- la **gestion des types** : vous ne pouvez plus modifier à la volée les types de variables, cela évite les erreurs et incohérences sur des gros projets
- une **meilleure prise en charge de la POO** (Programmation Orientée Objet)

JavaScript est un **langage à typage dynamique** : les types sont déterminés lors de l'interprétation du langage.

JavaScript est de base orienté prototype : la **programmation orientée prototype** est une forme de programmation orientée objet sans classe.

Un prototype est un objet à partir duquel on crée de nouveaux objets.

Contrairement à JavaScript, **TypeScript propose un typage statique** : les types de variables sont précisées par le développeur ou bien au moment de la compilation avant l'exécution du programme.

Greta Lyon Métropole

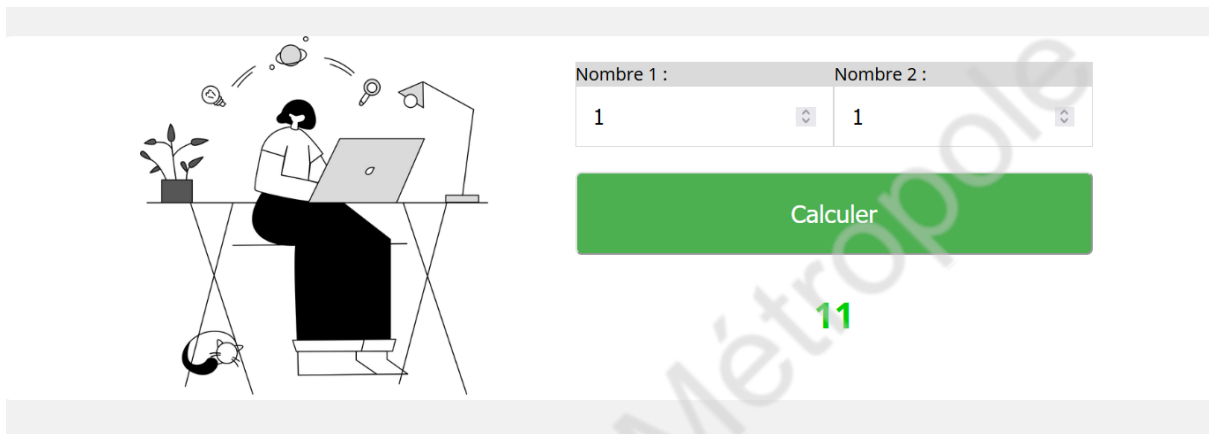
## 2. Avantages et inconvénients

Avantages	Inconvénients
<ul style="list-style-type: none"><li>• code robuste</li><li>• plus simple d'utilisation pour les développeurs une fois maîtrisé</li><li>• <b>les bugs sont détectés avant l'exécution du programme</b></li><li>• implémente ES6, des modules et des interfaces</li><li>• centré sur l'outillage</li></ul>	<ul style="list-style-type: none"><li>• ne peut pas être utilisé directement par les navigateurs ou NodeJS</li><li>• <b>le code doit être converti (compilé) dans le langage JavaScript avec le compilateur intégré de TypeScript TS</b></li></ul>

### 3. Exemple de problématique avec JavaScript

Dans un simple formulaire, le résultat de l'addition de nombres donne **par défaut un concaténation de strings**. Avec le symbole + en JavaScript vous pouvez aussi bien faire une concaténation qu'un calcul si les variables sont des nombres.

Vous devez faire une conversion de type avec l'une des méthodes `parseInt()`, `parseFloat()` ou `Number()` pour ne pas avoir d'incohérence.



📁 Intégrez le code ci-dessous ou récupérez le code source sur le dépôt Git.

index.html

```
<!DOCTYPE html>
<html lang="fr">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Additionner 2 nombres</title>
  <link rel="stylesheet" href="style.css">
</head>

<body>

  <section id="form-calcul" class="container">
```

```

<div class="bloc-img">
    
</div>

<form class="bloc-form">

    <div class="bloc-nombres">
        <div class="col">
            <label for="nb1">Nombre 1 :</label>
            <input type="number" name="nb1" id="nb1"
placeholder="Nombre 1">
        </div>

        <div class="col">
            <label for="nb2">Nombre 2 :</label>
            <input type="number" name="nb2" id="nb2"
placeholder="Nombre 2">
        </div>
    </div>

    <input type="button" value="Calculer" id="calcul">

    <div class="resultat"></div>

</form>

</section>

<script src="main.js"></script>

</body>

</html>

```

style.css

```

*, ::before, ::after{
    box-sizing: border-box;
    padding:0;
    margin:0;
}

```

```
html{
  font-size: 62,5%;
}

body{
  height:100vh;
  font-family: Open Sans, sans-serif;
  background: #F1F1F1;
  display:flex;
  justify-content: center;
  align-items: center;
}

.container{
  display:flex;
  justify-content: center;
  align-items: center;
  min-width:70vw;
  min-height:50vh;
  margin: 0 auto;
  background: white;
  border-radius: 1em;
  font-size:1rem;
}

.bloc-img{
  flex-basis:35%;
}

.bloc-img img{
  width:80%;
  height:80%;
  object-fit: cover;
  object-position: center;
}

.bloc-nombres{
  display:flex;
```

```
flex-direction: column;
background-color: #DADADA;
}

.bloc-form input[type="number"]{
  display: block;
  width:100%;
  padding: 0.8em;
  font-size:1.2em;
  border: 1px solid #DADADA;
  /*border: 1px solid rgb(0, 205, 10);*/
}

.bloc-form input[type="button"]{
  display: block;
  width:100%;
  background-color: #4CAF50;
  border-radius: 5px;
  margin-top:1em;
  color: white;
  padding: 1em;
  text-align: center;
  text-decoration: none;
  font-size: 1.4em;
}

.resultat{
  padding:1em;
  font-size:2em;
  font-weight: bolder;
  text-align: center;
  color:rgb(0, 205, 10);
}

@media all and (min-width: 600px){

  .bloc-nombres{
    flex-direction: row;
  }
}
```

main.js

```
const nb1 = document.querySelector("#nb1");
const nb2 = document.querySelector("#nb2");

document.querySelector("#calcul").addEventListener("click", function() {

    let resultat = addition(nb1.value, nb2.value);
    document.querySelector("#resultat").innerHTML = resultat;

});

function addition(n1,n2){
    return n1+n2;
}
```

 **Exercice : Comment corriger le code pour afficher un résultat cohérent sans modifier les constantes ?**

Vous pouvez utiliser une conversion de types avec `parseInt()` et utiliser des tests avec `typeof()` et `+nb1.value` pour convertir en nombre.



Nombre 1 :	Nombre 2 :
<input type="text" value="7"/>	<input type="text" value="5"/>

Calculer

12

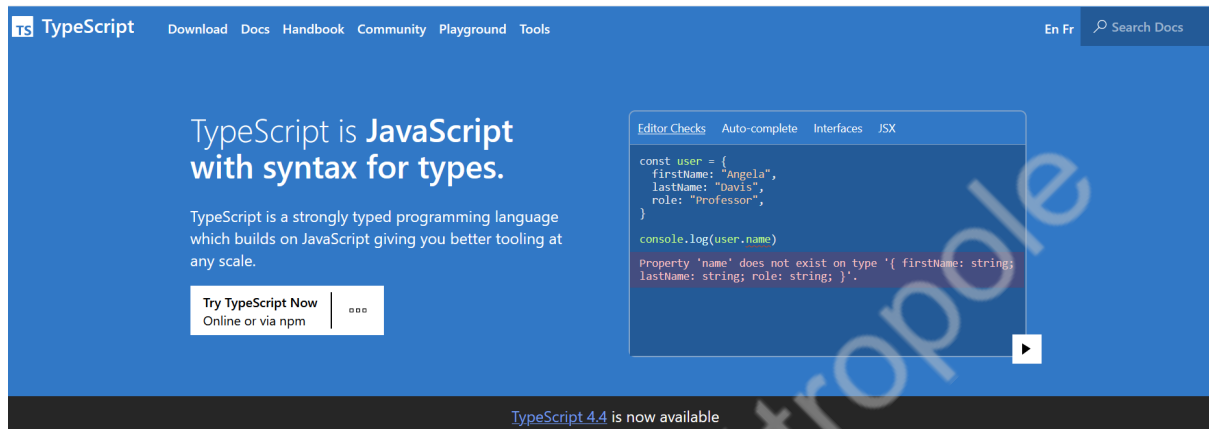
 Montrez la correction à votre formateur



## 4. Installer TypeScript

Site officiel

<https://www.typescriptlang.org/>



NodeJS doit être préalablement installé.

```
stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/exemple1/final
$ node -v
v14.15.1
```

Installez TypeScript **de façon globale** sur votre machine avec la console avec la commande suivante :

**npm install -g typescript**

```
stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/exemple1/final
$ npm install -g typescript

changed 1 package, and audited 2 packages in 7s

found 0 vulnerabilities
npm notice
npm notice New minor version of npm available! 7.20.6 -> 7.23.0
npm notice Changelog: https://github.com/npm/cli/releases/tag/v7.23.0
npm notice Run npm install -g npm@7.23.0 to update!
npm notice
```

De cette façon vous pourrez utiliser TypeScript directement sur tous vos projets.

Mettez à jour si nécessaire Node package manager npm

```
stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/exemple1/final
$ npm install -g npm@7.23.0

removed 223 packages, changed 5 packages, and audited 37 packages in 6s

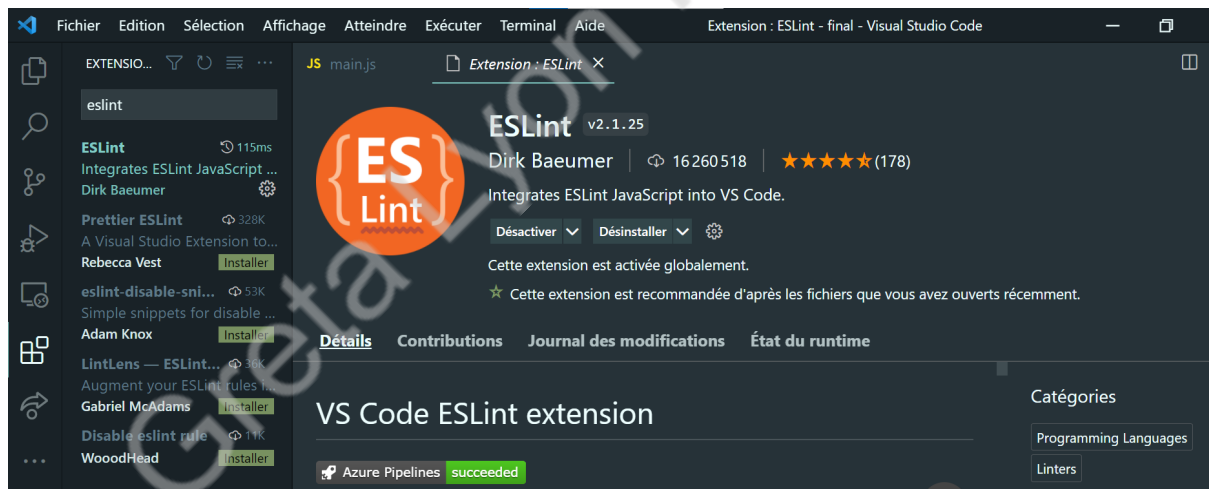
found 0 vulnerabilities
```

```
stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/exemple1/final
$ npm -v
7.23.0
```

## 5. Installer les extensions pour votre éditeur VSCode

### ESLint

Linteur pour Ecma Script : corrige et nettoie le code, trouve les bugs, ajoute l'auto-complétion.



### Material Icons

Sert à afficher les icônes selon le type de fichier dans l'explorateur.

**Material Icon Theme** v4.10.0  
 Philipp Kief | 8977228 | ★★★★★ (208)  
 Material Design Icons for Visual Studio Code

Définir le thème des icônes de fichier Désactiver Désinstaller ⚙️

Cette extension est activée globalement.

Détails	Contributions	Journal des modifications	État du runtime
Advpl_pnw	Denizenscript	Javaclass	Odin
Advpl_ptm	Dhall	Javascript	Opa
Advpl_ttp	Diff	Javascript-map	Opam
Android	Dinophp	Jenkins	Pascal
Angular	Disc	Jest	Pawn
Angular-component	Django	Jinja	Pdf
Angular-directive	Docker	Jsconfig	Percy
Angular-guard	Document	Json	Port
			Snowpack
			Snyk
			Solidity
			Stencil
			Stitches
			Storybook
			Stryker
			Stylelint

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL

## Path Intellisense

Gère l'auto-complétion pour les chemins de fichiers.

**Path Intellisense** v2.4.0  
 Christian Kohler | 5360230 | ★★★★★ (90)  
 Visual Studio Code plugin that autocompletes filenames

Désactiver Désinstaller ⚙️

Cette extension est activée globalement.

**Détails** Contributions Journal des modifications État du runtime

### Installation

In the command palette (cmd-shift-p) select Install Extension and choose Path Intellisense.

To use Path Intellisense instead of the default autocompletion, the following configuration option must be added to your settings:

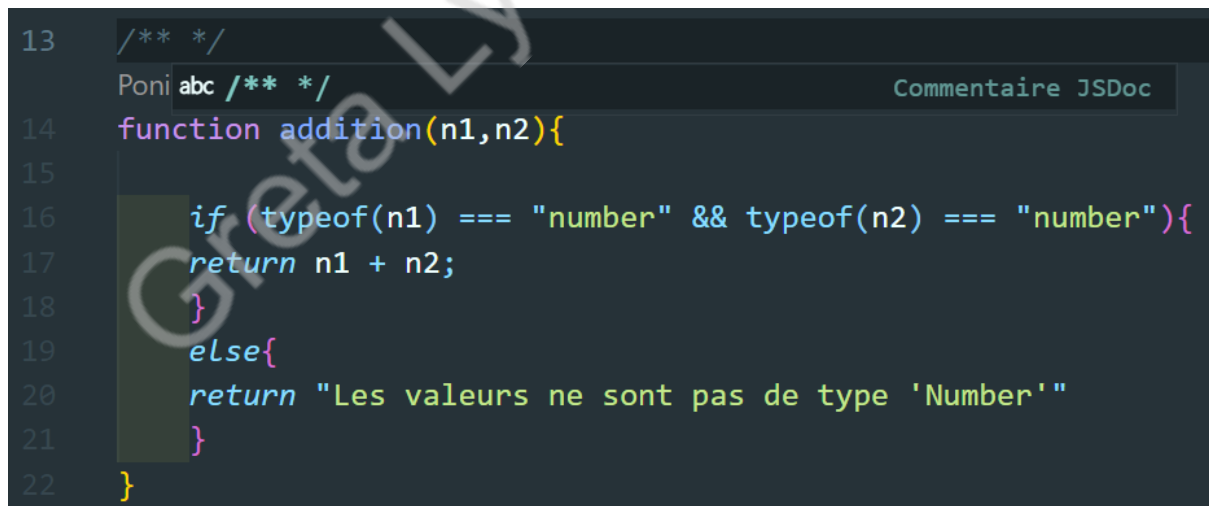
```
{ "typescript.suggest.paths": false }
```

## Ad JSDoc Comment

Sert à documenter votre code JavaScript.



Tapez la ligne de commentaire `/** */` avant votre fonction ou bien la commande F1 sur Windows pour afficher le terminal et tapez Add doc comments puis entrée.



```
/**
 *
 * @param {*} n1
 * @param {*} n2
 * @returns
 */
function addition(n1,n2){

    if (typeof(n1) === "number" && typeof(n2) === "number"){
        return n1 + n2;
    }
    else{
        return "Les valeurs ne sont pas de type 'Number'"
    }
}
```

## 6. Réécrire le JavaScript en TypeScript

Ajoutez un dossier TypeScript/exemple1/

Placez le code source de l'exemple précédent à l'intérieur

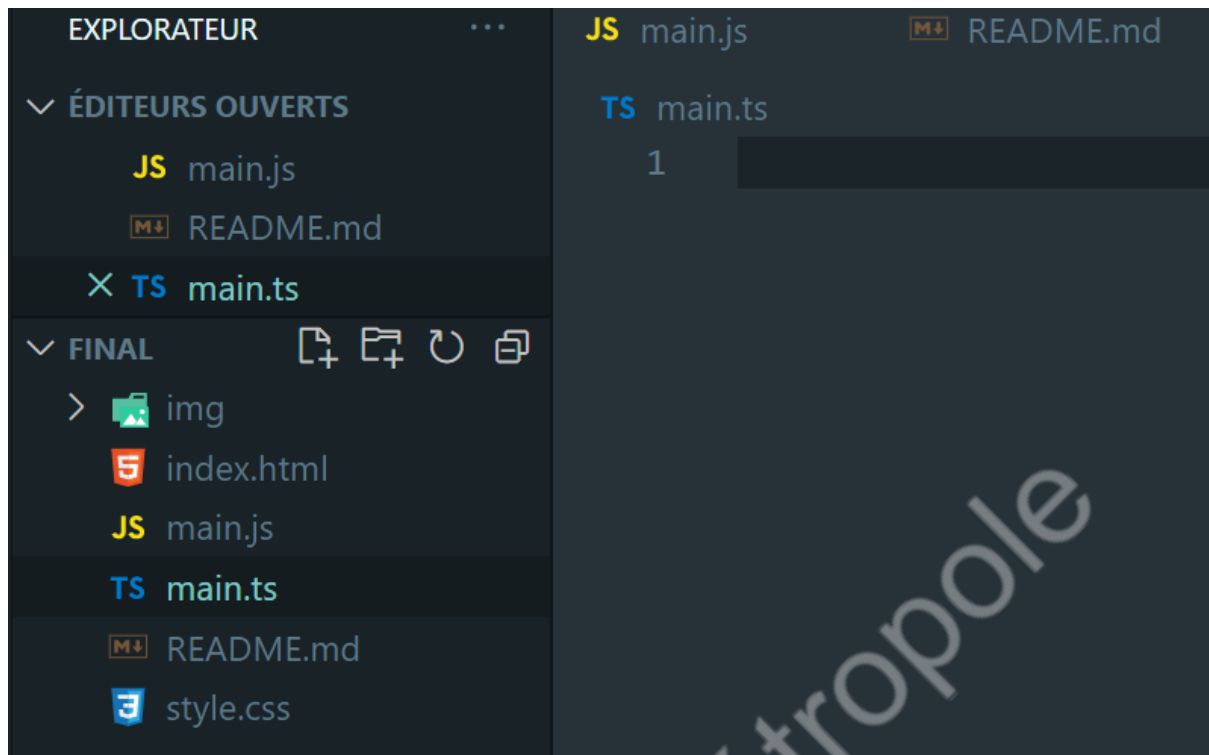
### Ajouter un nouveau fichier main.ts

Créez un nouveau fichier avec l'extension .ts.

Copiez le code du fichier main.js dans ce nouveau fichier.

Greta Lyon Métropole

```
JS main.js  README.md  TS main.ts 6 X
TS main.ts > ...
1  const nb1 = document.querySelector("#nb1");
2  const nb2 = document.querySelector("#nb2");
3
4  document.querySelector("#calcul").addEventListener("click", () =
5
6      let resultat = addition(+nb1.value, +nb2.value);
7
8      document.querySelector(".resultat").innerHTML = resultat;
9
10 });
11
12
13 Ponicode: Flash Test | Ponicode: Unit Test
14 function addition(n1,n2){
15     if (typeof(n1) === "number" && typeof(n2) === "number"){
16         return n1 + n2;
17     }
18     else{
19         return "Les valeurs ne sont pas de type 'Number'"
20     }
21 }
22
23
```



Supprimez le fichier main.js.

## Compilez main.ts

Dans la console tapez :

**tsc main.ts**

Un rapport d'erreur apparaît dans le terminal intégré. Un nouveau fichier main.js a été automatiquement généré.



```

stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/exemple1/final
$ tsc main.ts
main.ts:6:34 - error TS2339: Property 'value' does not exist on type 'Element'.

6   let resultat = addition(+nb1.value, +nb2.value);
                                ~~~~~

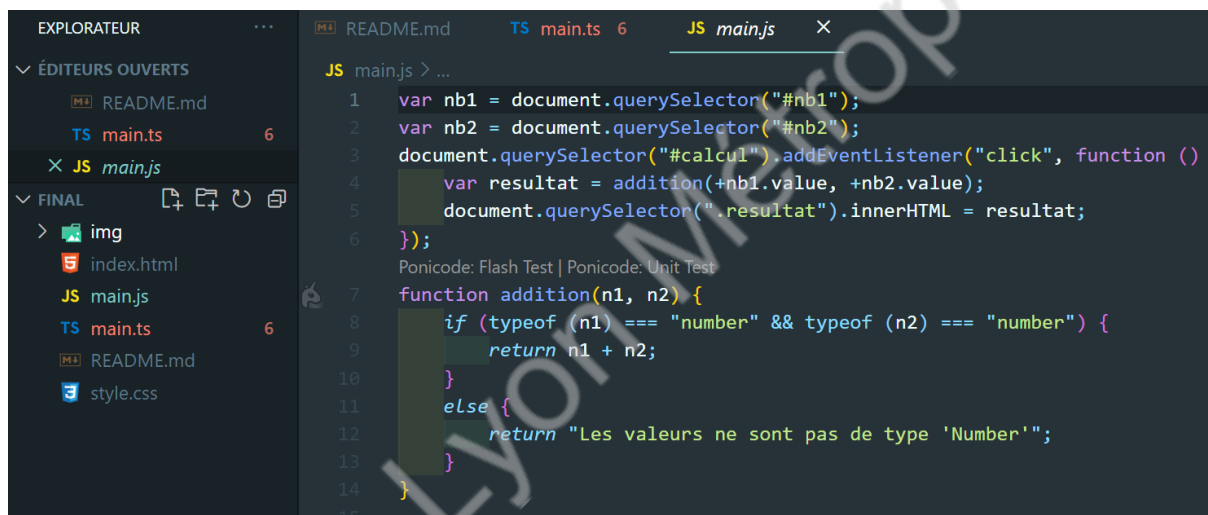
main.ts:6:46 - error TS2339: Property 'value' does not exist on type 'Element'.

6   let resultat = addition(+nb1.value, +nb2.value);
                                ~~~~~

main.ts:8:5 - error TS2322: Type 'string | number' is not assignable to type 'string'.
  Type 'number' is not assignable to type 'string'.

8   document.querySelector(".resultat").innerHTML = resultat;
    ~~~~~

```



La syntaxe est modifiée en ES5 (ancienne version de JavaScript standard).

Dans VSCode désactivez le fichier main.js : vous travaillerez avec le fichier main.ts.

## Typage les variables

Pour corriger les erreurs qui apparaissent dans le fichier main.ts, nous indiquons le typage pour chaque variable

```

const nb1 = document.querySelector("#nb1") as HTMLInputElement;
const nb2 = document.querySelector("#nb2") as HTMLInputElement;

```

Lorsque vous survolez la variable, le type s'affiche dans une infobulle.

```
TS main.ts > [?] const nb1: HTMLInputElement
1  const nb1 = document.querySelector("#nb1") as HTMLInputElement;
2  const nb2 = document.querySelector("#nb2") as HTMLInputElement;
```

Créez une nouvelle variable divResultat en choisissant le type HTMLDivElement

```
const nb1 = document.querySelector("#nb1") as HTMLInputElement;
const nb2 = document.querySelector("#nb2") as HTMLInputElement;
const divResultat = document.querySelector(".resultat") as
HTMLDivElement;
```

Lorsque vous réaffectez divResultat une erreur s'affiche toujours.

```
const nb1 = document.querySelector("#nb1") as HTMLInputElement;
const nb2 = document.querySelector("#nb2") as HTMLInputElement;
const divResultat = document.querySelector(".resultat") as
HTMLDivElement;

document.querySelector("#calcul").addEventListener("click", () =>{

    let resultat = addition(+nb1.value, +nb2.value);
    divResultat.innerHTML = resultat;
});
```

Si vous passez votre souris sur la ligne, il est indiqué "qu'il est impossible d'assigner le type number au type string".

```
5 Impossible d'assigner le type 'string | number' au type 'string'. ;
  Impossible d'assigner le type 'number' au type 'string'. ts(2322)
u
const divResultat: HTMLDivElement
Voir le problème Aucune solution disponible dans l'immédiat
divResultat.innerHTML = resultat;
```

Vous pourriez corriger en ajoutant une chaîne de caractère : ""+resultat

```
document.querySelector("#calcul").addEventListener("click", () =>{

    let resultat = addition(+nb1.value, +nb2.value);
```

```
divResultat.innerHTML = "" + resultat;  
});
```

Sinon

```
const nb1 = document.querySelector("#nb1") as HTMLInputElement;  
const nb2 = document.querySelector("#nb2") as HTMLInputElement;  
const divResultat = document.querySelector(".resultat") as  
HTMLDivElement;  
  
document.querySelector("#calcul").addEventListener("click", () =>{  
  
    let resultat = addition(+nb1.value, +nb2.value);  
    divResultat.innerHTML = resultat.toString();  
});  
  
function addition(nb1,nb2){  
  
    if (typeof(nb1) === "number" && typeof(nb2) === "number"){  
        return nb1 + nb2;  
    }  
    else{  
        return "Les valeurs ne sont pas de type 'Number'"  
    }  
}
```

Relancez la commande tsc main.ts

```
stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/exemple1/final  
$ tsc main.ts  
  
stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/exemple1/final  
$
```

Effectuez un nouveau test dans le navigateur :



Nombre 1 :	Nombre 2 :
Nombre 1 <input type="text"/>	Nombre 2 <input type="text"/>

Calculer



Nombre 1 :	Nombre 2 :
5 <input type="text"/>	2 <input type="text"/>

Calculer

7

## 7. Autre façon d'indiquer le typage

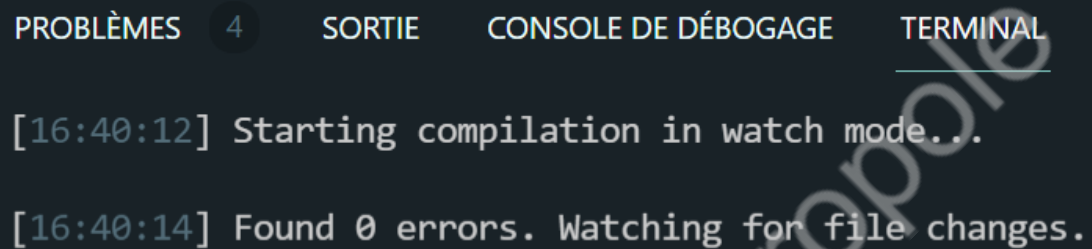
```
const nb1 = <HTMLInputElement>document.querySelector("#nb1");  
const nb2 = <HTMLInputElement>document.querySelector("#nb2");  
const divResultat =  
<HTMLDivElement>document.querySelector(".resultat");
```

Greta Lyon Métropole

## 8. Vérifier en continu la compilation

Si vous lancez la commande suivante, le fichier est automatiquement mis à jour lors de vos modifications sans avoir à relancer la commande de compilation.

**tsc main --watch** ou bien **tsc main -w**



The screenshot shows a terminal window with a dark background. At the top, there are four tabs: 'PROBLÈMES' (with a blue circle containing the number 4), 'SORTIE', 'CONSOLE DE DÉBOGAGE', and 'TERMINAL' (which is underlined). The terminal output shows two lines of text: '[16:40:12] Starting compilation in watch mode...' and '[16:40:14] Found 0 errors. Watching for file changes.' A yellow cursor is visible on the line following the second message. A large, light gray watermark 'Greta Lyon Métropole' is diagonally across the terminal area.

```
PROBLÈMES 4  SORTIE  CONSOLE DE DÉBOGAGE  TERMINAL  
  
[16:40:12] Starting compilation in watch mode...  
  
[16:40:14] Found 0 errors. Watching for file changes.  
|
```

## 9. Simplifier la fonction

Il vous suffit de spécifier le type attendu dans les paramètres de la fonction

```
function addition(nb1:number,nb2:number) {
    return nb1 + nb2;
}
```

```
const nb1 = document.querySelector("#nb1") as HTMLInputElement;
const nb2 = document.querySelector("#nb2") as HTMLInputElement;
const divResultat = document.querySelector(".resultat") as
HTMLDivElement;

document.querySelector("#calcul").addEventListener("click", () =>{

    let resultat = addition(+nb1.value, +nb2.value);
    divResultat.innerHTML = resultat.toString();
});

function addition(nb1:number,nb2:number) {
    return nb1 + nb2;
}
```

Avec ce script vous ne pouvez envoyer que des nombres. Pour la fonction addition vous obligez l'utilisation de number.

Définir un type pour vos données, ou pour la valeur de retour de vos fonctions, vous permet de comprendre plus rapidement le fonctionnement du code.

### Code source complet final

```
const nb1 = <HTMLInputElement>document.querySelector("#nb1");
const nb2 = <HTMLInputElement>document.querySelector("#nb2");
const divResultat =
<HTMLDivElement>document.querySelector(".resultat");

document.querySelector("#calcul").addEventListener("click", () =>{
```

```
    let resultat = addition(+nb1.value, +nb2.value);  
    divResultat.innerHTML = resultat.toString();  
  });  
  
function addition(nb1:number,nb2:number){  
  return nb1 + nb2;  
}
```

Greta Lyon Métropole

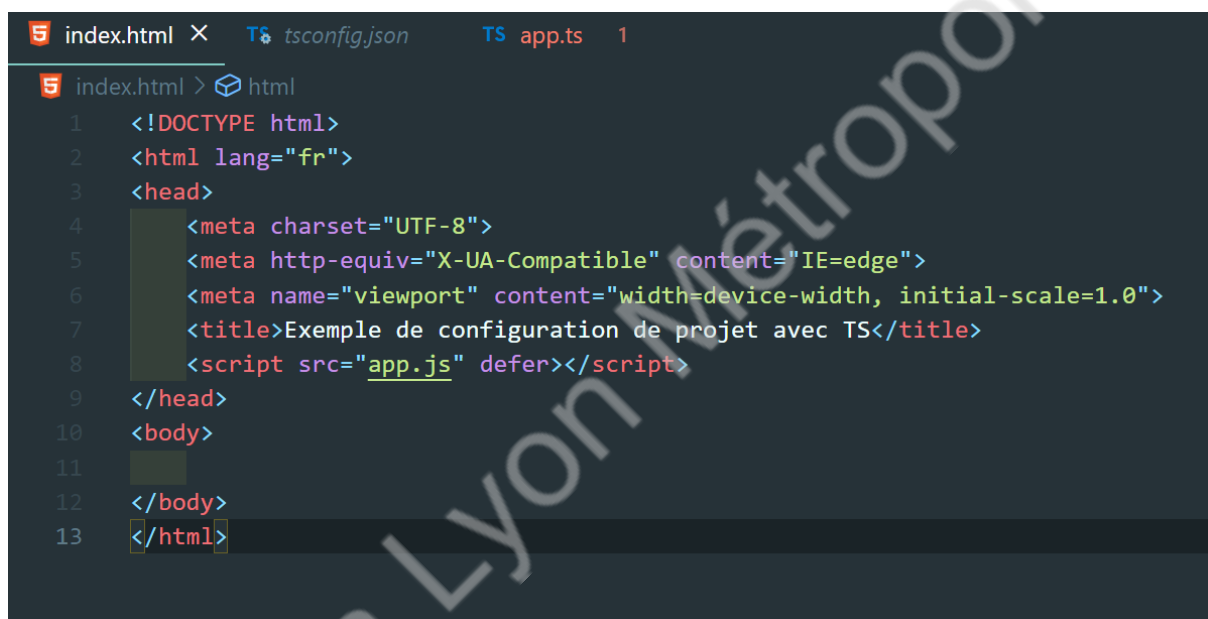


## 10. Ajouter TypeScript à un projet existant ou préparer un projet

### Création du dossier

Créez un nouveau dossier et placez un fichier index.html.

Dans le fichier index.html ajoutez le lien vers le script app.js



```
index.html X TS tsconfig.json TS app.ts 1
index.html > html
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Exemple de configuration de projet avec TS</title>
8   <script src="app.js" defer></script>
9 </head>
10 <body>
11
12 </body>
13 </html>
```

Ajoutez un fichier **app.ts** dans le dossier. Lorsque vous aurez à compiler, le code sera transformé en code JavaScript dans un fichier app.js.

**Placer une balise d'appel à notre application dans le fichier index.html**

```
<!DOCTYPE html>
<html lang="fr">
<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Exemple de configuration de projet avec TS</title>
  <script src="app.js" defer></script>
</head>
<body>
  <div id="app"></div>
</body>
</html>
```

Dans votre fichier app.ts, récupérez votre balise div qui comporte l'identifiant #app

```
const app = <HTMLDivElement>document.querySelector("#app");
```

## Placer du contenu dans l'application

app.ts

```
const app = <HTMLDivElement>document.querySelector("#app");

app.innerHTML="<h1>Exemple</h1><main>Coucou !</main>"
```

## Initialiser le projet

Ouvrez le terminal intégré et lancez la commande **tsc --init**

Un fichier **tsconfig.js** est placé à la racine du projet.

Ce fichier au format JSON indique des **options de compilation**.

```

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 */
8      // "tsBuildInfoFile": ".",      /* Specify the folder for .tsbuildi */
9      // "disableSourceOfProjectReferenceRedirect": true, /* Disable preferring source files */
10     // "disableSolutionSearching": true, /* Opt a project out of multi-project */
11     // "disableReferencedProjectLoad": true, /* Reduce the number of projects loaded */
12
13     /* Language and Environment */
14     "target": "es5",                /* Set the JavaScript language version */
15     // "lib": [],                   /* Specify a set of bundled library */
16     // "jsx": "preserve",           /* Specify what JSX code is generated */
17     // "experimentalDecorators": true, /* Enable experimental support for */
18     // "emitDecoratorMetadata": true, /* Emit design-type metadata for */
19     // "jsxFactory": "",            /* Specify the JSX factory function */
20     // "jsxFragmentFactory": "",    /* Specify the JSX Fragment reference */

```

Certaines propriétés sont déjà préremplies.

La **propriété target** indique la sortie du code en ES5, ES6...

La **propriété lib** pourra nous servir à renseigner des librairies.

Ce fichier documente tous les autres éléments de configuration que vous pouvez utiliser et activer au besoin de votre projet.

Si vous activez cette propriété lib, vous serez obligé de renseigner les librairies de base, ajoutées par défaut.

```

/* Language and Environment */
"target": "es5",
"lib": ["DOM", "DOM.Iterable", "ScriptHost", "ES6"],

```

**Créer un fichier app.ts au même niveau dans le dossier et générer le fichier app.js**

tsc app.ts

```

EXPLOREUR
└─ Éditeurs ouverts
  ├─ index.html
  ├─ tsconfig.json
  └─ app.ts
└─ PROJET-EXEMPLE
  ├─ app.js
  ├─ app.ts
  ├─ index.html
  └─ tsconfig.json

index.html
1 <!DOCTYPE html>
2 <html lang="fr">
3 <head>
4   <meta charset="UTF-8">
5   <meta http-equiv="X-UA-Compatible" content="IE=edge">
6   <meta name="viewport" content="width=device-width, initial-scale=1.0">
7   <title>Exemple de configuration de projet avec TS</title>
8   <script src="app.js" defer></script>
9 </head>
10 <body>

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL
stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/projet-exemple
$ tsc app.ts

stephane@LAPTOP-D8S5TIQ7 MINGW64 ~/Desktop/Sandbox/TypeScript/projet-exemple
$

```

## Le mode strict

Le fichier généré en JavaScript par TypeScript ajoute en première instruction “use strict”.

```

JS app.js > ...
1 "use strict";
2 var app = document.querySelector("#app");
3 app.innerHTML = "<h1>Exemple</h1><main>Coucou !</main>";
4

```

Ce mode change le comportement de JavaScript. Certaines variables non déclarées avec `let` ou `const` (non scoppées) sont par exemple interdites.

Le mode strict peut s’appliquer pour le scope de fonctions.

Avec `const` et `let` la portée (scope) est limitée à celle du bloc dans lequel sont déclarées les variables.

Cela impose d’être un meilleur développeur et de coder proprement. **Cela évite des erreurs silencieuses.**

Ouvrez le projet dans votre navigateur :

# Exemple

Coucou !

## Exclure des fichiers de la compilation

Créez un nouveau fichier test.ts

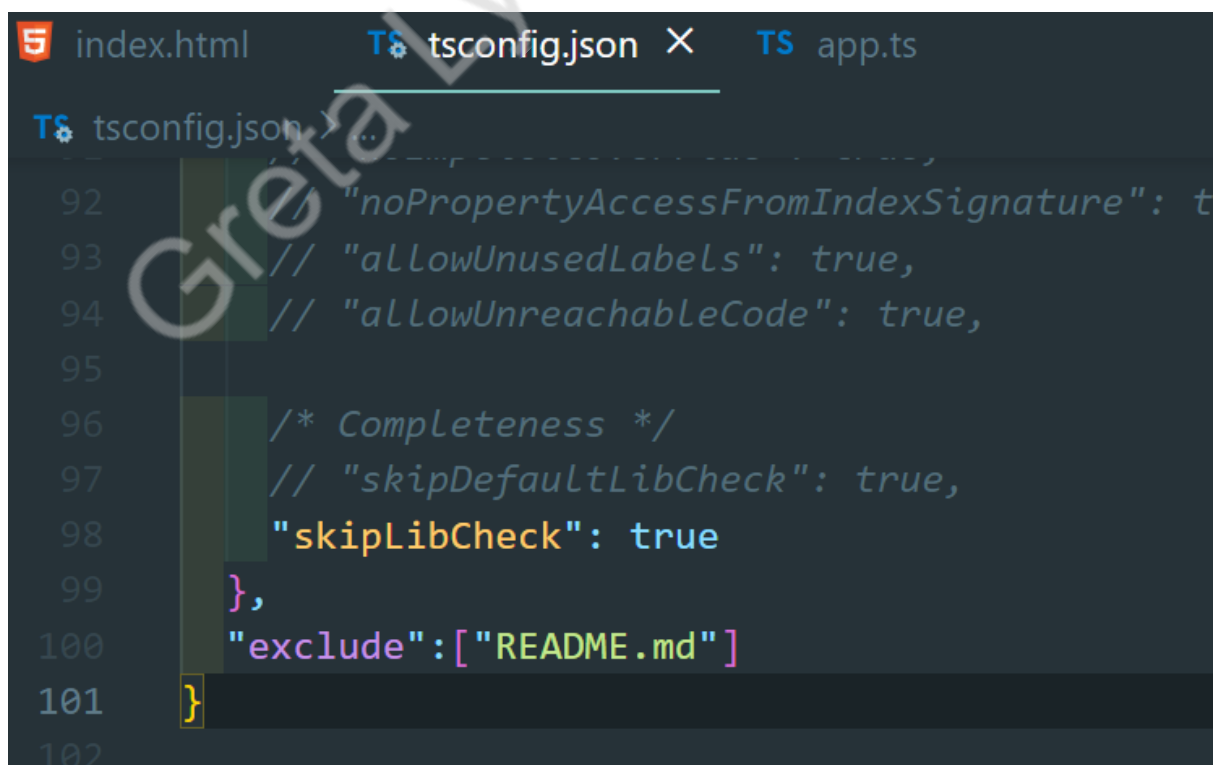
A la fin du fichier tsconfig.json ajoutez dans une propriété exclude, le fichier à exclure dans un tableau.

Vous pouvez ajouter autant de fichiers que vous voulez en les séparant par des virgules.



The image shows a file explorer on the left with a project named 'PROJET-EXEMPLE' containing files: app.js, app.ts, index.html, test.ts, and tsconfig.json. The main editor shows the tsconfig.json file with the following content:

```
94 // "allowUnreachableCode": true,  
95  
96 /* Completeness */  
97 // "skipDefaultLibCheck": true,  
98 "skipLibCheck": true  
99 },  
100 "exclude": ["test.ts", ]  
101 }
```



The image shows a code editor with tabs for index.html, tsconfig.json, and app.ts. The tsconfig.json file is open, showing the following content:

```
92 // "noPropertyAccessFromIndexSignature": true,  
93 // "allowUnusedLabels": true,  
94 // "allowUnreachableCode": true,  
95  
96 /* Completeness */  
97 // "skipDefaultLibCheck": true,  
98 "skipLibCheck": true  
99 },  
100 "exclude": ["README.md"]  
101 }
```

Vous pourriez par exemple exclure un dossier node-modules

```
},  
"exclude":["test.ts","node_modules"]  
}
```

A contrario vous pouvez aussi utiliser la propriété "include".

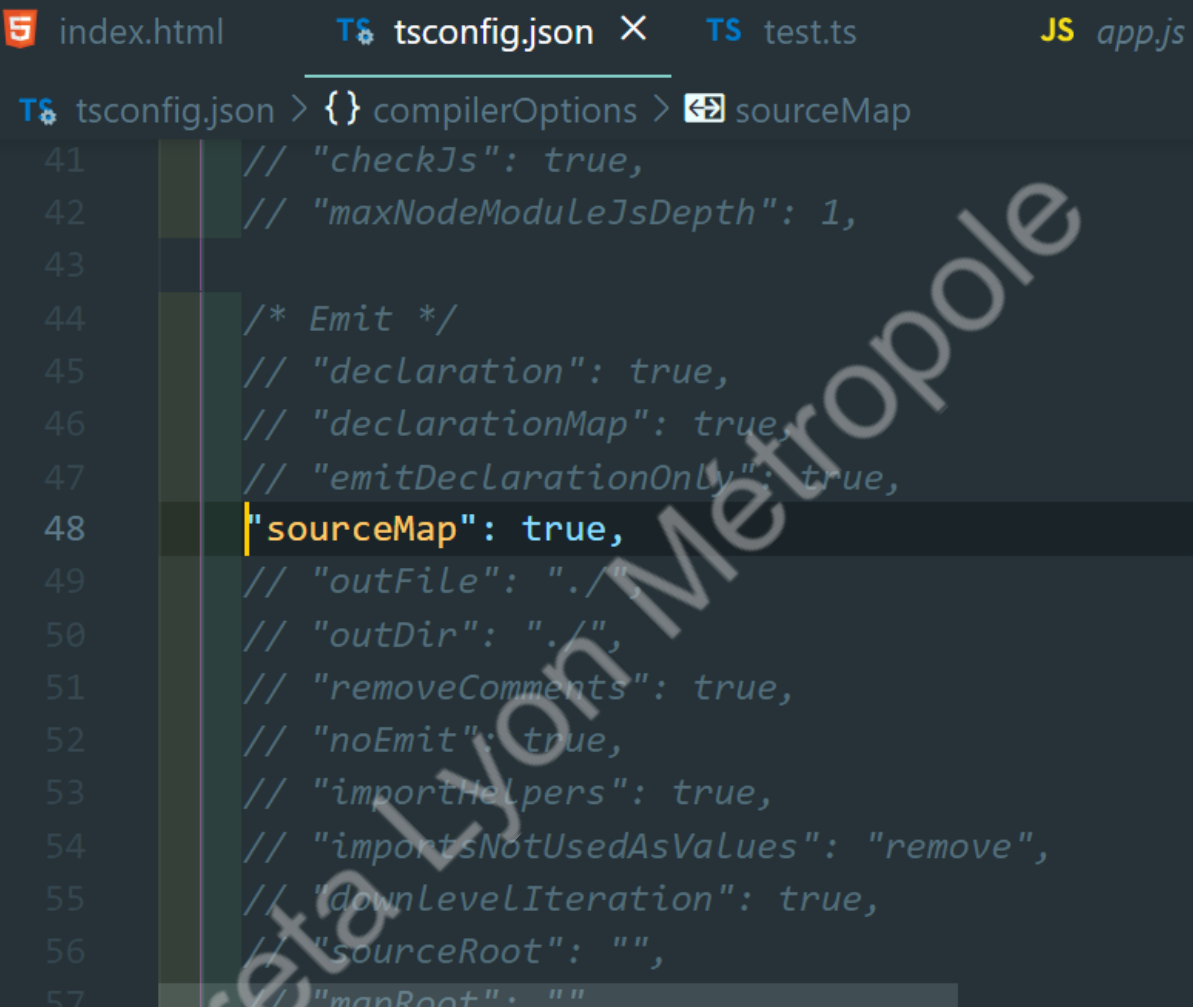
## Compiler tous les fichiers TS du projet

Saisissez la commande **tsc -w**

Greta Lyon Métropole

## 11. Activer le mode debug de TypeScript dans Chrome

Dans le fichier tsconfig.js du projet activez la propriété **sourceMap**

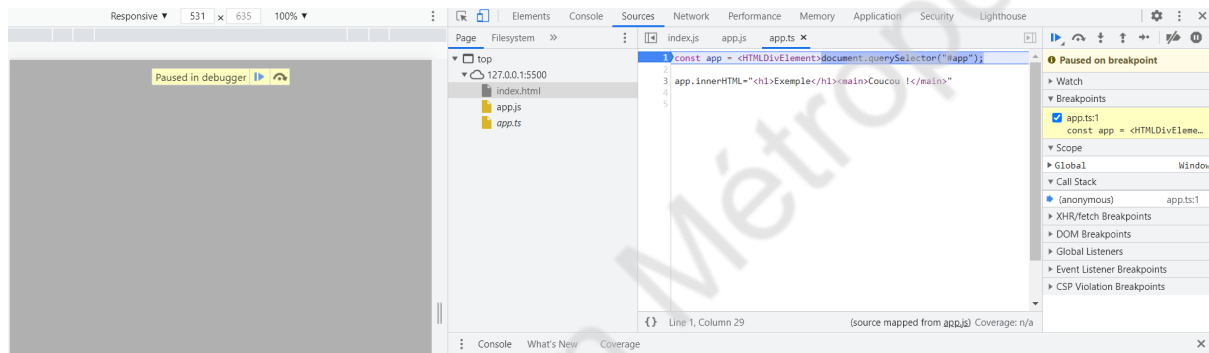
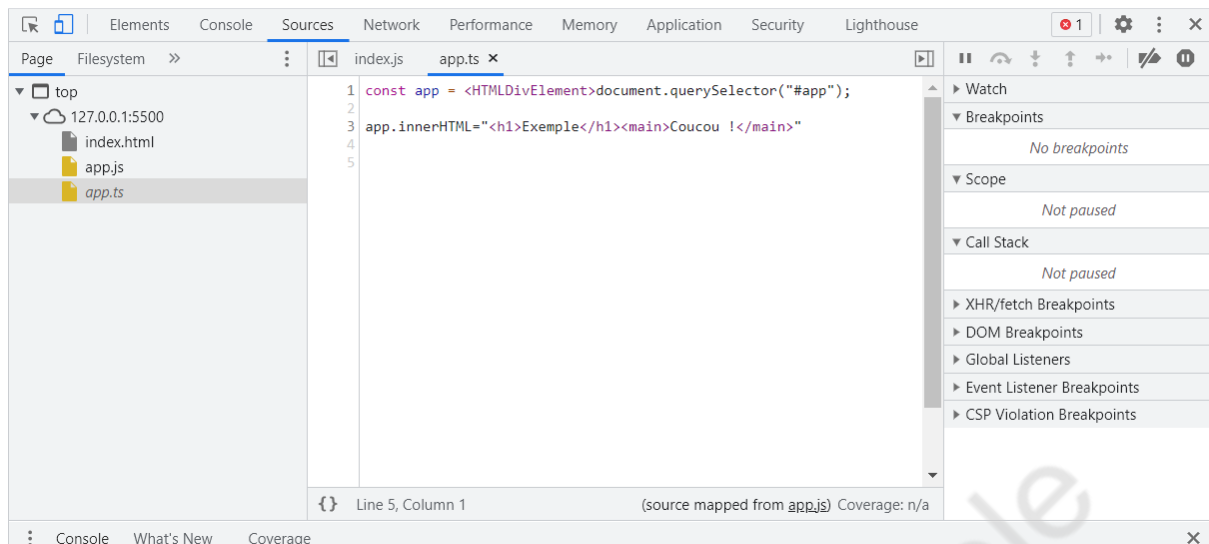


```
index.html  TS tsconfig.json X  TS test.ts  JS app.js
TS tsconfig.json > {} compilerOptions > sourceMap
41 // "checkJs": true,
42 // "maxNodeModuleJsDepth": 1,
43
44 /* Emit */
45 // "declaration": true,
46 // "declarationMap": true,
47 // "emitDeclarationOnly": true,
48 "sourceMap": true,
49 // "outFile": "./",
50 // "outDir": "./",
51 // "removeComments": true,
52 // "noEmit": true,
53 // "importHelpers": true,
54 // "importsNotUsedAsValues": "remove",
55 // "downlevelIteration": true,
56 // "sourceRoot": "",
57 // "mapRoot": ""
```

Ouvrez ensuite la page index.html avec votre navigateur Chrome.

F12 ou Ctrl Maj I pour ouvrir les outils du développeurs.

Cliquez sur l'onglet Sources pour voir le fichier app.ts caché par défaut.



Cliquez sur la première ligne de votre fichier app.ts pour ajouter un point d'arrêt.

Rafraîchissez votre navigateur. Visualisez étape par étape en appuyant sur les boutons de flèches haut et bas pour déboguer.

## 12. Bien structurer un projet

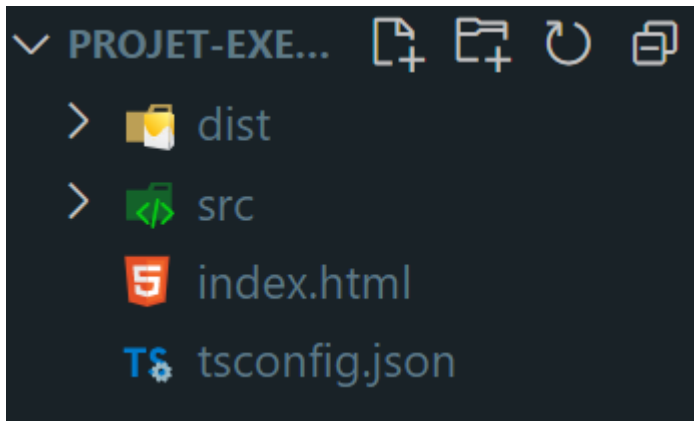
### Dossier source src

Dans ce dossier placez vos fichiers source \*.ts

### Dossier de distribution dist

Ce dossier contient les fichiers \*.js à intégrer dans les pages web





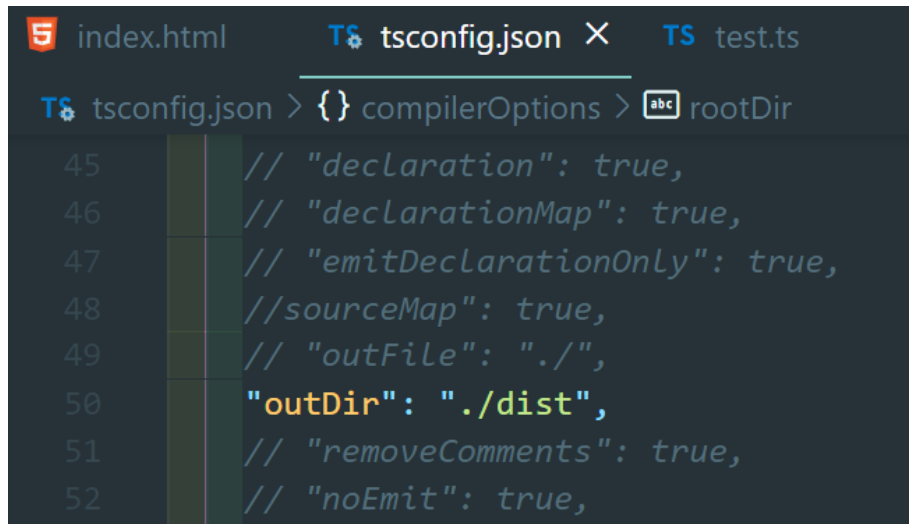
## Spécifier les chemins dans le fichier tsconfig.json

Enlevez les commentaires sur les propriétés **rootDir** (source) de même que pour **outDir** (distribution)



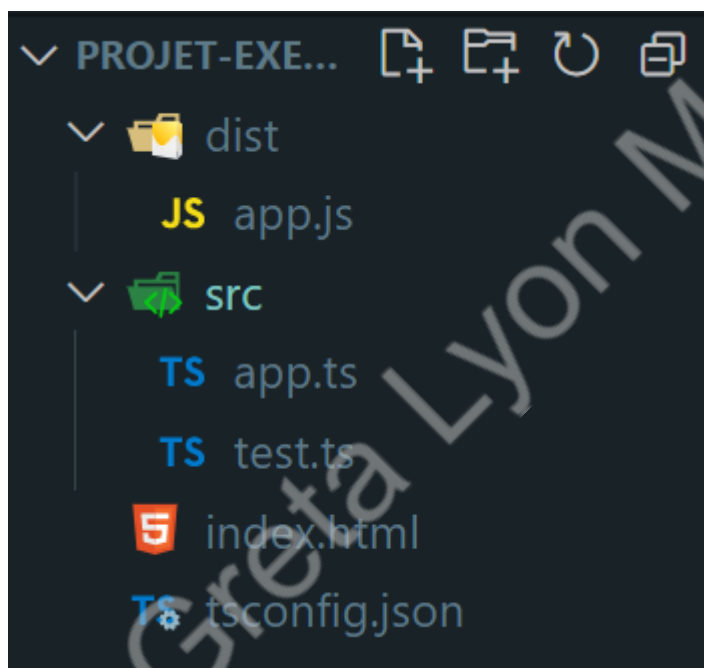
Changez le chemin :

```
"rootDir": "./src",
```



```
index.html  TS tsconfig.json X  TS test.ts
TS tsconfig.json > {} compilerOptions > rootDir
45 // "declaration": true,
46 // "declarationMap": true,
47 // "emitDeclarationOnly": true,
48 // "sourceMap": true,
49 // "outFile": "./",
50 "outDir": "./dist",
51 // "removeComments": true,
52 // "noEmit": true,
```

Vue du dossier projet :



## 13. En synthèse

Le développeur ne s'occupe plus que des fichiers .ts. Les fichiers .js sont interprétés par le navigateur.

Installation du compilateur TypeScript : **npm install -g typescript**

Vérifier l'installation : **tsc -v**

Transformer un code TypeScript en JavaScript : **tsc nomdufichier.ts**

Superviser le changement de code : **tsc nomdufichier.ts -w**

TypeScript est généralement considéré comme une **bonne pratique** par la communauté des développeurs Web.

## 14. Ressources

TypeScript : concepts de base, Programmation Orientée Objet, concepts avancés, mise en pratique et projets - Matthieu GASTON H2Prog 2021

TypeScript Notions fondamentales - Felix BILLON Sylvain PONTOREAU ENI 2019

Dépôt GitHub : <https://github.com/microsoft/TypeScript>

Qu'est-ce que le mode strict ? [https://www.youtube.com/watch?v=4kk-Dee6O\\_g](https://www.youtube.com/watch?v=4kk-Dee6O_g)

Documentation TsConfig : <https://www.typescriptlang.org/docs/handbook/tsconfig-json.html>