# Comment exécuter TypeScript dans WebStorm

by Aphinya Dechalert | Posted on September 30, 2021

WebStorm est un IDE de JetBrains qui se targue d'être "l'IDE JavaScript le plus intelligent" du marché. Mais l'est-il vraiment ? Et que peut faire exactement WebStorm pour vous ?

WebStorm fait partie de la suite d'IDE de JetBrains spécifiques à un langage. Cet outil spécialisé se concentre sur le développement JavaScript et dispose d'une suite complète d'outils de développement intégrés, d'une navigation rapide, d'une fonction de recherche, d'environnements personnalisables et d'intégrations de travail en équipe en temps réel. L'une des principales caractéristiques de WebStorm est sa prise en charge du code source TypeScript.

WebStorm reconnaît les fichiers .ts et .tsx avec des systèmes de support de code pour rendre votre flux de travail aussi transparent que possible. Pour les développeurs JavaScript, en particulier ceux qui travaillent avec Angular, le support de TypeScript dans un IDE peut faire ou défaire le plafond de verre de la productivité. Voici un guide rapide sur la façon d'exécuter TypeScript dans WebStorm et de développer votre première application pour vous aider à démarrer.

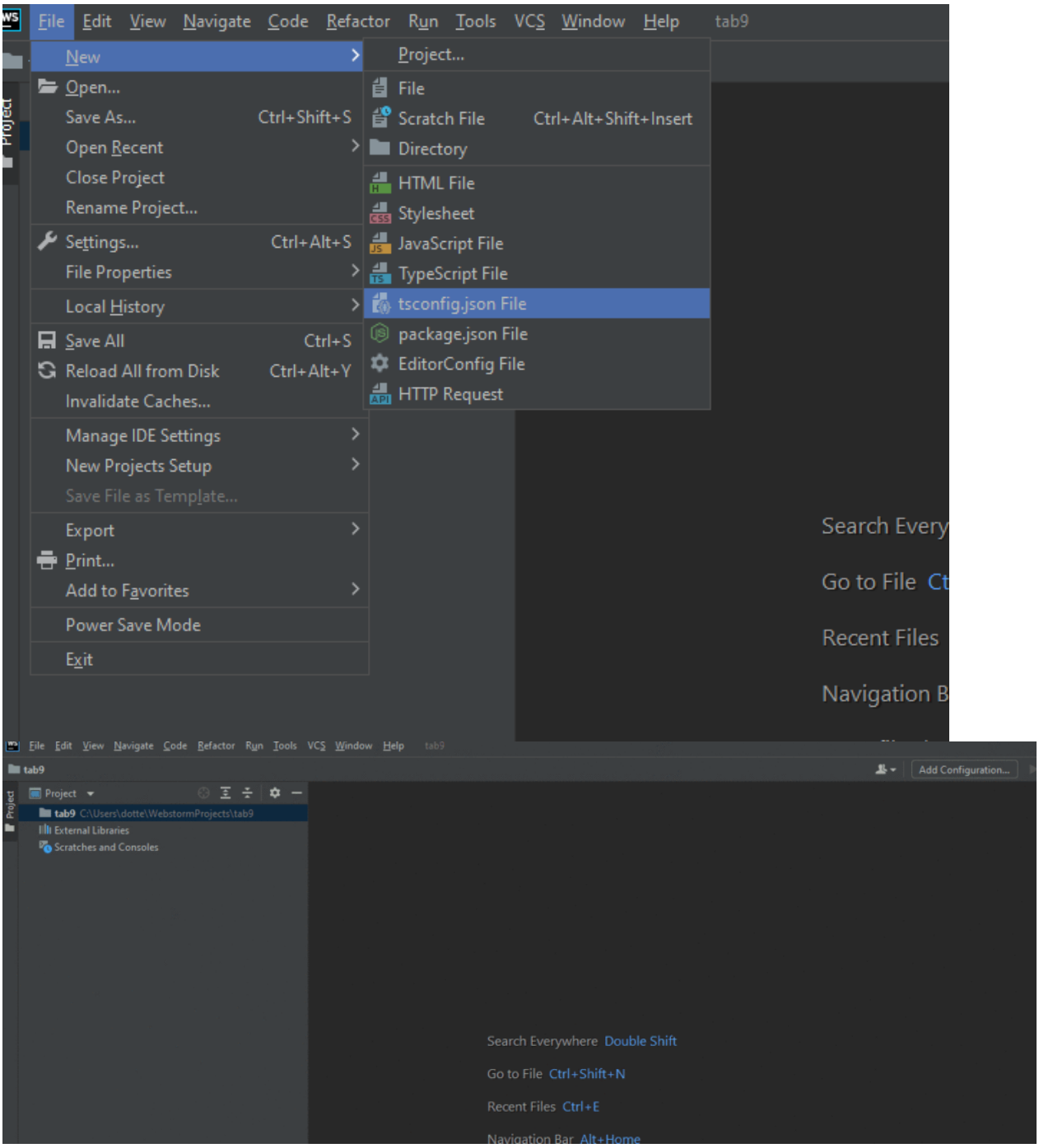
## Installation et configuration de TypeScript

Lorsqu'il s'agit de TypeScript avec WebStorm, vous n'avez pas grand-chose à faire. En effet, WebStorm est livré avec des modèles de projets intégrés que vous pouvez utiliser dès le départ.

Lorsque vous démarrez WebStorm, vous avez la possibilité de choisir un projet vide ou une configuration standard pour les bibliothèques et les cadres JavaScript les plus populaires tels que Angular, Cordova, Express, Meteor, Node.js, React, React Native et Vue.js.

Si vous créez un nouveau projet à partir de zéro et que vous avez besoin d'un fichier tsconfig.json, le modèle de base peut facilement être démarré en allant à l'adresse suivante

**File** > **New** > **tsconfig.json File**.




Vous obtiendrez un fichier tsconfig.json par défaut qui ressemble à quelque chose comme ceci :

```
{
  “compilerOptions”: {
    “module”: “commonjs”,
    “target”: “es5”,
    “sourceMap”: true
  },
  “exclude”: [
    “node_modules”
  ]}
```

**Pour configurer votre projet TypeScript, la convention est d'avoir un dossier src et un dossier build. Le dossier src est l'endroit où tout votre code ts sera placé, tandis que le dossier build est l'endroit où vos sorties de compilation JavaScript seront compilées.**

Une fois cette opération terminée, vous devez ajouter "outDir" : "build" à vos compilerOptions. Le build à outDir correspond au dossier de build que vous avez créé. En théorie, cela peut être tout ce que vous voulez, tant que les noms correspondent les uns aux autres.

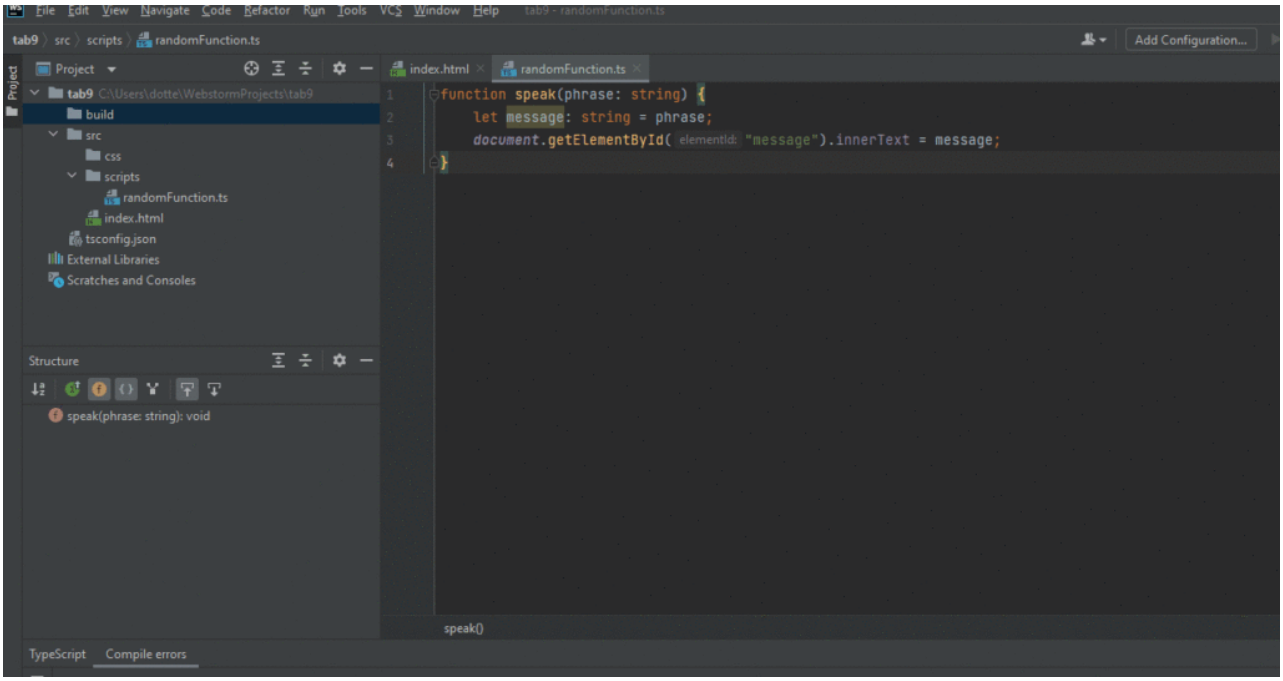
Pour indiquer à TypeScript quel dossier vous voulez construire, ajoutez le chemin d'accès au dossier à inclure. Voici un exemple de la configuration finale du fichier TypeScript tsconfig.json avec les répertoires build et src :

```
{
  "compilerOptions": {
    "module": "commonjs",
    "target": "es5",
    "sourceMap": true,
    "outDir": "build"
  },
  "exclude": [
    "node_modules"
  ],
  "include": [
    "src/scripts/**/*"
  ]
}
```

# Compiler TypeScript dans WebStorm

Maintenant que les bases sont posées, vous pouvez commencer à créer votre projet TypeScript dans WebStorm. L'étape suivante consistera à compiler TypeScript dans WebStorm.

Le plus simple est de faire un clic droit n'importe où dans votre espace de travail TypeScript, et tout en bas de la liste qui s'affiche, il y a une option Compile TypeScript. Une fois que vous avez cliqué sur cette option, votre code TypeScript sera compilé automatiquement dans WebStorm.




Il est bon de noter que pour que TypeScript compile correctement, l'inclusion dans votre tsconfig.json doit correspondre au chemin du fichier. Dans la capture d'écran ci-dessus, le chemin du fichier se trouve dans src/scripts. Notre partie include correspond à ce chemin :

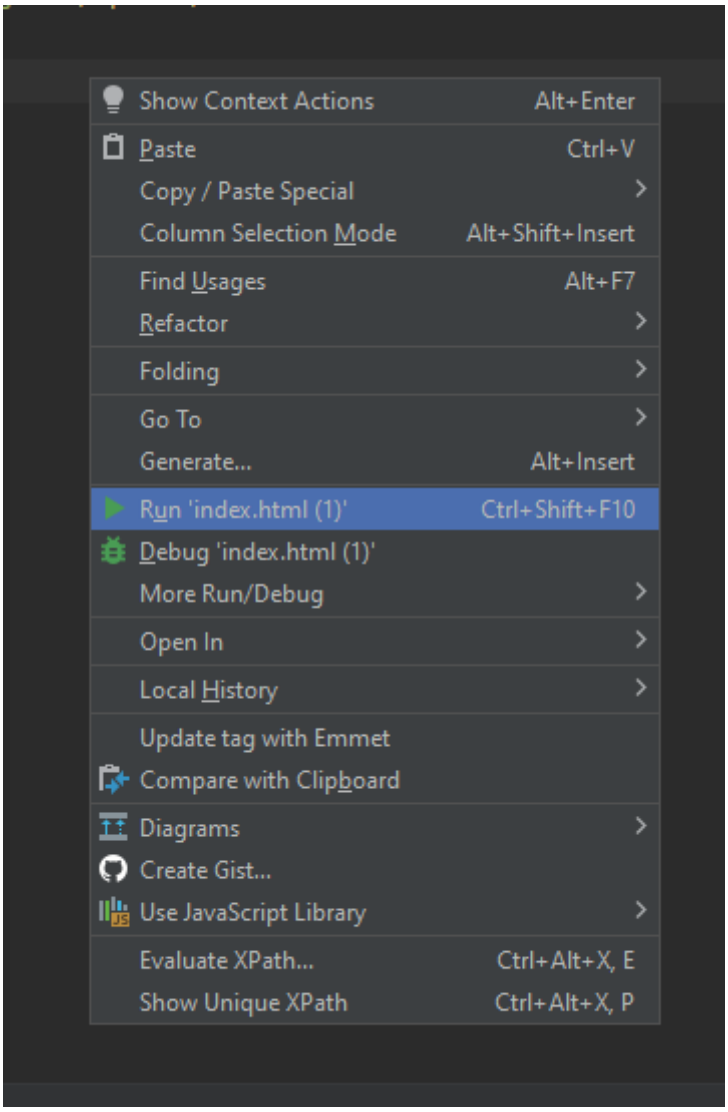
…

```
"include": [
  "src/scripts/**/*"
]
```

…

# Exécuter votre projet TypeScript sur WebStorm

Il existe plusieurs façons d'exécuter votre TypeScript. L'une d'entre elles consiste à sélectionner Run.




Vous pouvez également utiliser la combinaison de commandes clavier Ctrl + Shift + F10. Ces deux options feront apparaître une page localhost sur votre navigateur par défaut.

Toutefois, si vous souhaitez tester sur un autre navigateur, vous pouvez sélectionner l'une des icônes de navigateur situées en haut à droite de votre écran - comme indiqué en rouge ci-dessous.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>TypeScript</title>
    <script src="../build/randomFunction.js"></script>
</head>
<body>

<h1>Running TypeScript on WebStorm</h1>

<button id="totalButton"
        onclick="speak('Hi there TabNine!')">say something</button>
<div><span id="message"></span></div>




</body>
</html>
```

Voici une capture d'écran du localhost généré à partir du code ci-dessus.


Une fois que vous avez installé votre serveur local, WebStorm vous met automatiquement en place le rechargement à chaud. Cela signifie que vous n'avez pas à intervenir avec un rafraîchissement de la page chaque fois que vous modifiez votre code.

# Déboguer TypeScript dans WebStorm

À un moment donné, vous voudrez déboguer votre code. Bien que console.log() puisse vous aider à retrouver des éléments, les outils de débogage de WebStorm peuvent accélérer le processus sans encombrer votre code. De plus, utiliser console.log() pour déboguer n'est pas la meilleure pratique.

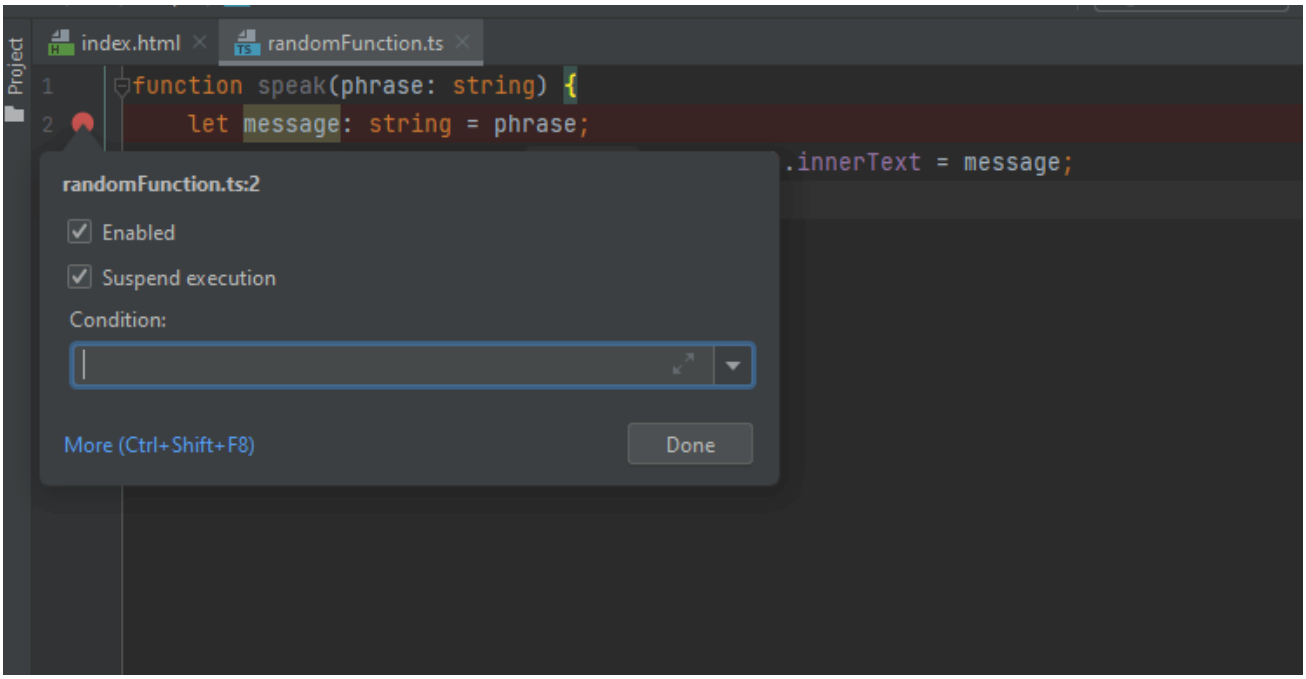
Bien que console.log puisse aider dans certains cas particuliers, la plupart du temps, ce dont vous avez besoin, c'est que votre code ralentisse. Cela vous permet d'assimiler mentalement ce qui se passe et de déterminer précisément ce qui se passe, pourquoi cela se produit et si cela contribue au bogue que vous essayez de corriger.

**En ce qui concerne TypeScript, vous travaillez très probablement sur une application à page unique (SPA) ou une application côté serveur basée sur node.js. Cela signifie que vous avez également affaire à des choses comme les cycles de vie. L'utilisation de points d'arrêt vous permet de mettre votre code en pause afin d'avoir une chance de voir l'impact ligne par ligne pour chaque région du code.**

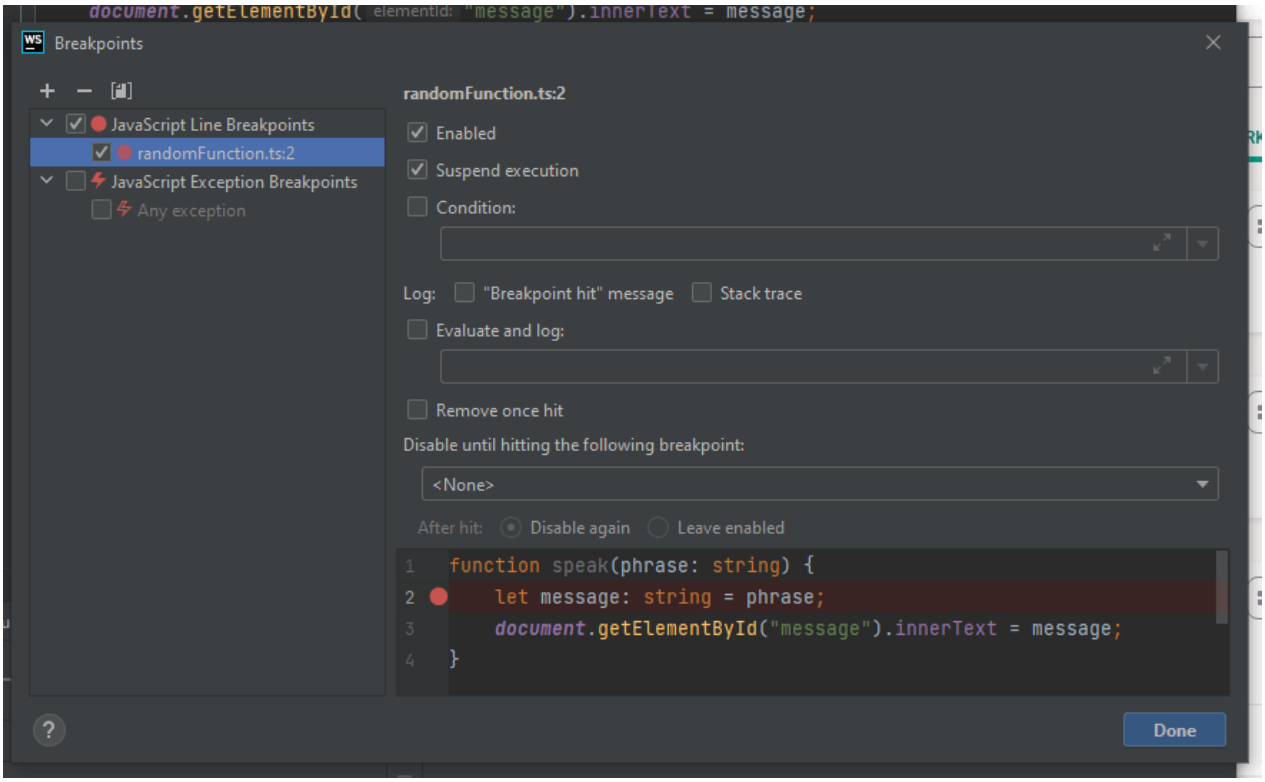
Pour déboguer TypeScript dans WebStorm, vous pouvez créer des points d'arrêt dans votre code en cliquant sur le numéro de ligne. Cela produira un point rouge, où le code sera mis en pause une fois qu'il aura atteint le point d'arrêt.


Si vous cliquez avec le bouton droit de la souris sur le point d'arrêt rouge, vous pouvez également le configurer avec des conditionnels, des journaux, un suivi de pile et le désactiver en fonction des exceptions. Ces fonctions peuvent s'avérer très utiles lorsque vous souhaitez affiner votre processus de débogage dans WebStorm for TypeScript.

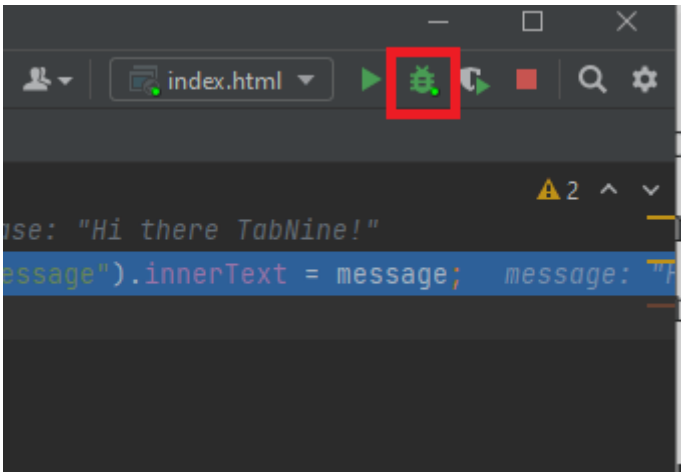
Voici une capture d'écran de ce à quoi il ressemble :



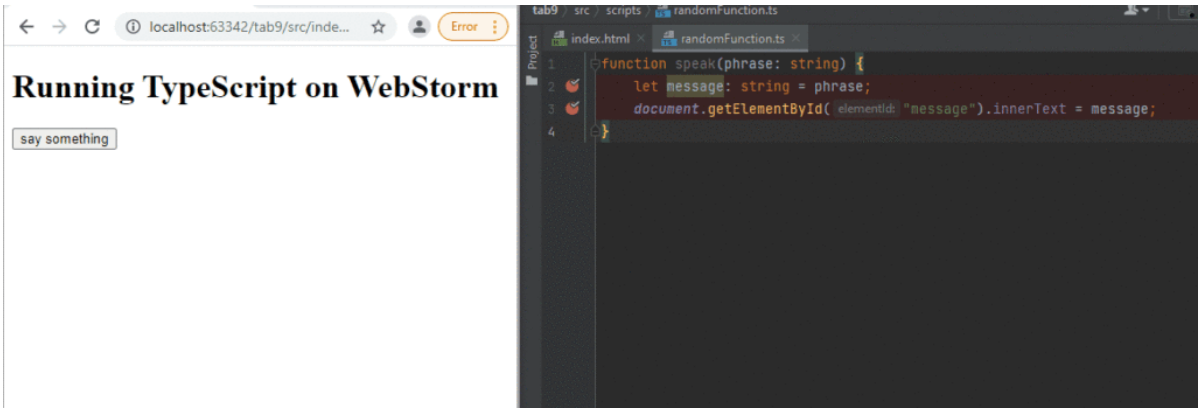

Pour faire apparaître d'autres options, cliquez sur More ou Ctrl + Shift + F8. Voici le reste de ce que WebStorm propose pour le débogage de TypeScript.




Pour lancer le débogueur, cliquez sur la petite icône verte de l'insecte. Cela fera apparaître une nouvelle fenêtre localhost, dans laquelle vous pourrez tester le flux de processus de votre code.




Voici un exemple de ce à quoi ressemble le débogueur TypeScript de WebStorm en action avec des points d'arrêt :

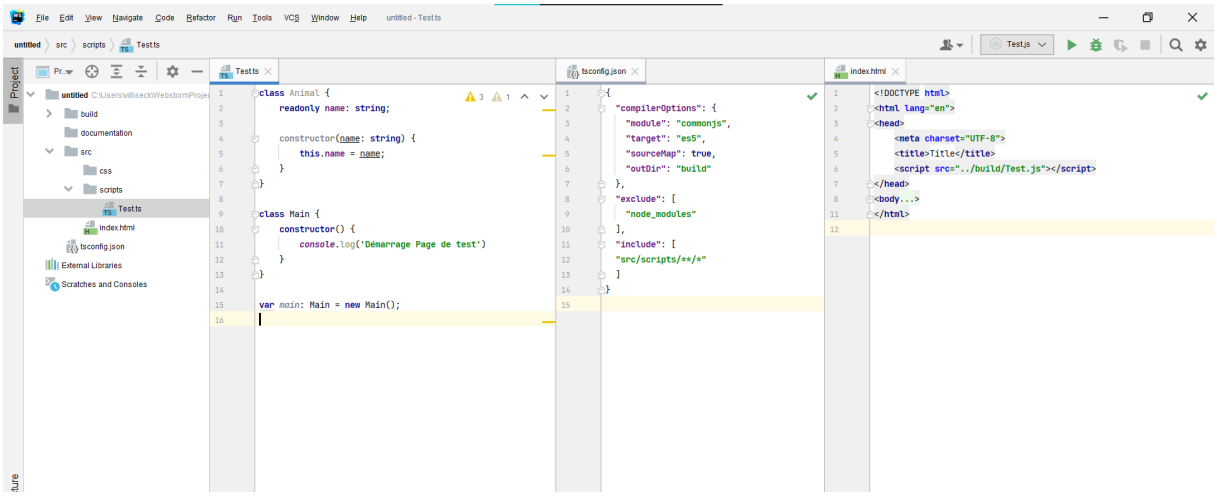



# Synthèse

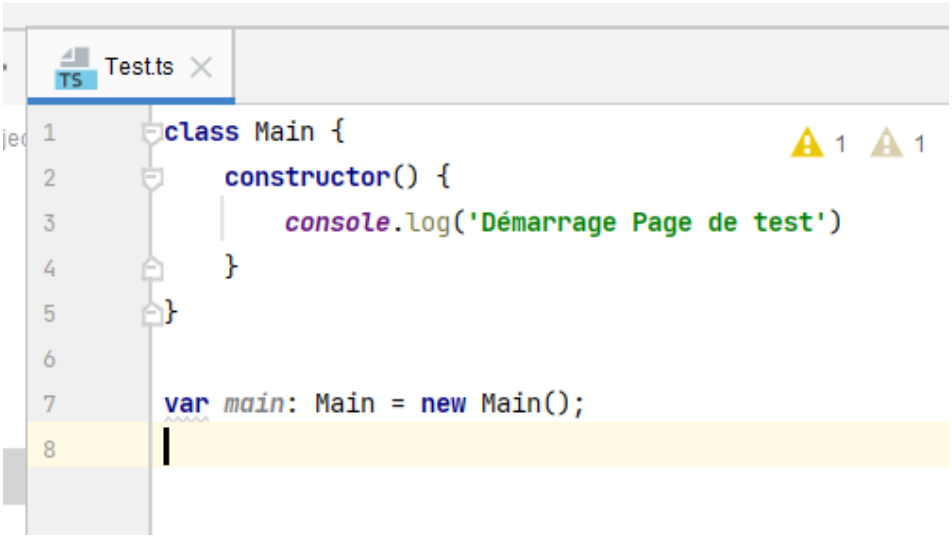
Lorsqu'il s'agit de TypeScript, WebStorm fait le gros du travail à votre place. Il n'y a qu'un minimum d'installation - à part la configuration de votre projet. L'exécution de TypeScript dans WebStorm est un jeu d'enfant, sans qu'il soit nécessaire de s'occuper des lignes de commande, de la configuration des serveurs locaux ou des trop nombreuses étapes de la compilation de TypeScript en code JavaScript. Tout est livré prêt à l'emploi et entièrement optimisé pour les projets TypeScript. Le débogage est intégré par défaut, vous n'avez donc pas besoin de le configurer ou de télécharger des plugins supplémentaires pour commencer. Pour TypeScript dans WebStorm, les choses fonctionnent.

Globalement, si vous voulez une expérience transparente, essayez WebStorm pour votre prochain projet TypeScript et constatez la différence dans votre flux de codage. Bien que ce ne soit pas un IDE gratuit, le coût de l'abonnement vaut les avantages qu'il apporte à votre capacité à produire du code rapidement.

## Fichier ts de départ avec le main pour démarrer le code




Zoom




Equivaut à la classe Main en java (point d'entrée de l'application)