

# **Introduction et installation de TypeScript**

# Table des matières

<b>I. Contexte</b>	<b>3</b>
<b>II. L'historique</b>	<b>3</b>
A. Historique .....	3
B. Exercice : Quiz .....	8
<b>III. L'installation de TypeScript</b>	<b>9</b>
A. Installation de TypeScript .....	9
B. Exercice : Quiz .....	12
<b>IV. Essentiel</b>	<b>13</b>
<b>V. Auto-évaluation</b>	<b>13</b>
A. Exercice .....	13
B. Test .....	14
<b>Solutions des exercices</b>	<b>15</b>

## I. Contexte

**Durée : 1 heure**

**Prérequis : les bases en JavaScript**

**Environnement de travail : VS Code**

### Contexte

Dans ce cours, nous allons découvrir ensemble le langage de programmation TypeScript. Nous allons voir tout ce que vous devez savoir pour créer votre première application en TypeScript.

Pour ce faire, nous commencerons par un bref historique, en mettant l'accent sur la raison d'être de TypeScript. Nous allons faire une comparaison avec son ancêtre JavaScript, en mettant en avant leurs avantages et leurs limites.

Une bonne base de connaissances en JavaScript est un atout pour aborder TypeScript, étant donné que ces 2 langages sont pratiquement identiques. Si vous n'avez aucune expérience en JavaScript, ce n'est pas un problème, vous pouvez tout de même suivre ce cours. TypeScript est un langage à part entière qui peut être abordé indépendamment de JavaScript. Néanmoins, si vous avez de bonnes bases en JavaScript, apprendre le langage TypeScript sera encore plus facile.

Par la suite, nous procéderons à l'installation étape par étape, en mettant au clair la fonction de chaque commande.

Ensuite, nous aborderons quelques exemples de TypeScript. Nous verrons également comment convertir un code TypeScript vers du JavaScript. Nous allons mettre l'accent sur les différences qui séparent ces 2 langages.

### Attention

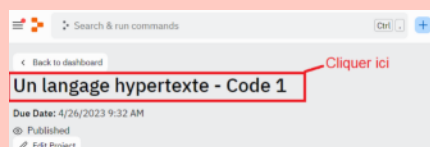
Pour avoir accès au code et à l'IDE intégré de cette leçon, vous devez :

- 1) Vous connecter à votre compte sur <https://replit.com/> (ou créer gratuitement votre compte)
- 2) Rejoindre la Team Code Studi du module via ce lien : <https://replit.com/teams/join/mmurvlgippxuasordloklllbqskoim-programmer-avec-javascript>

Une fois ces étapes effectuées, nous vous conseillons de rafraîchir votre navigateur si le code ne s'affiche pas.

En cas de problème, redémarrez votre navigateur et vérifiez que vous avez bien accepté les cookies de connexion nécessaires avant de recommencer la procédure.

Pour accéder au code dans votre cours, cliquez sur le nom du lien Replit dans la fenêtre. Par exemple :



## II. L'historique

### A. Historique

#### Définition

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

TypeScript est un *superset* ou un *sur-ensemble* de JavaScript, ce qui veut dire que JavaScript est un sous-ensemble de TypeScript. Pour être plus précis, TypeScript est comme le JavaScript, sauf qu'il possède un typage en plus. On peut aussi le définir comme une version améliorée de JavaScript.

*Il a été élu The 2nd most loved programming language sur StackOverflow developer survey en 2020<sup>1</sup>, et troisième des Most Wanted Programming Language en 2022<sup>2</sup>.*

TypeScript a été développé en 2010 par Microsoft, afin de pallier les limitations qu'avait JavaScript lors du développement de projets d'envergure. Il a d'abord été utilisé pour des projets internes à Microsoft avant d'être disponible mondialement en 2012. Depuis, de nouvelles versions sont sorties. En 2013 sort TypeScript 0.9, TypeScript 1.0 a été publié en 2014, puis, TypeScript 2.0 en 2016, TypeScript 3.0 a été publié en 2018, la version 4 est sortie en février 2021. La version la plus récente est la version 5 sortie en mars 2023.

Ceci est un bref historique pour vous remettre dans le contexte. Maintenant, nous allons comprendre l'intérêt d'apprendre ce langage. Qu'a-t-il de si particulier ? JavaScript n'est-il pas mieux ?

### Pourquoi apprendre TypeScript ?

JavaScript est de loin le langage le plus utilisé si on en croit les statistiques. Il possède une communauté très forte, et existe depuis plusieurs années que ce soit sur les navigateurs ou encore sur les serveurs. Alors pourquoi utiliser TypeScript ?

En effet, JavaScript a beaucoup de mérites, cependant, il a aussi beaucoup de **lacunes**, notamment pour le développement de grands projets.

Pourquoi les grands projets ? Les grands projets demandent plus de rigueur, et ça, c'est le talon d'Achille de JavaScript. De plus, pour la cohésion des équipes qui travaillent sur le projet, documenter le code tout au long de développement n'est pas automatique sur JavaScript. De plus, JavaScript est trop permissif et difficile à déboguer. Il est tout de même important de rappeler que JavaScript reste le langage de programmation le plus couramment utilisé pour le développement d'application. De plus, il est nécessaire de compiler les applications développées en TypeScript vers JavaScript pour pouvoir les exécuter, ce qui rend ce langage incontournable.

Comment TypeScript peut-il pallier ces lacunes ?

TypeScript est considéré comme une version améliorée de JavaScript, ce qui n'est pas faux, étant donné qu'il élimine les défauts cités précédemment, tout en gardant ce qu'il y a de meilleur dans JavaScript.

Voici une liste non exhaustive des **avantages** de TypeScript :

- Compatibilité parfaite avec JavaScript : il suffit de changer l'extension .ts en .js pour avoir un fichier JavaScript, et vice-versa.
- TypeScript peut être transpilé en JavaScript : ce qui veut dire que tout ce que JavaScript peut faire, TypeScript peut le faire aussi.
- Toutes les bibliothèques JavaScript peuvent être utilisées pour TypeScript.
- Il est possible de créer une bibliothèque avec TypeScript et de l'exporter pour l'utiliser avec JavaScript.
- TypeScript peut aussi être flexible comme son homologue JavaScript : en ajoutant un typage dynamique.
- Couche de sécurité avec le typage : à l'instar de Java et C++, TypeScript est un langage fortement typé, ce qui lui fait gagner en popularité auprès de la communauté Java et C++.
- TypeScript est un langage multi-paradigmes (nous pouvons faire de la programmation fonctionnelle, comme de la programmation orientée objet).

### Exemple TypeScript Vs JavaScript

Comme mentionné précédemment, connaître JavaScript n'est pas une obligation, cependant, avoir des bases en JavaScript vous sera d'une grande utilité. Si nous regardons bien le code ci-dessous, nous comprendrons pourquoi.

1 <https://insights.stackoverflow.com/survey/2020#most-loved-dreaded-and-wanted>

2 <https://survey.stackoverflow.co/2022/#technology-most-loved-dreaded-and-wanted>

```
1 //Objet TypeScript
2 let client = {
3     nom : "Aubergine",
4     tel : 7777777777
5 }
6 console.log(client.nom);
```

[cf. ]

```
1 //Objet JavaScript
2 let client = {
3     nom : "Aubergine",
4     tel : 7777777777
5 };
6 console.log(client.nom);
```

[cf. ]

Dans le cas de déclaration d'objets comme ci-dessus, il n'y a aucune différence entre TypeScript et JavaScript, mis à part le point-virgule. Dans la déclaration des variables, c'est aussi identique. Voir l'exemple ci-dessous :

```
1 //TypeScript (typé)
2 let stud1 = "Lola";
3 let note1 = 13;

1 //JavaScript (non-typé)
2 let stud1 = "Lola";
3 let note1 = 13;
```

JavaScript est un langage **dynamique**, ce qui signifie qu'il est très souple dans la façon dont il vous permet de définir et d'utiliser les variables. Dans l'exemple ci-dessous, nous avons déclaré 2 variables sans assigner de type comme sur TypeScript.

Ce qui veut dire que vous pouvez décider de créer une variable et de lui affecter une chaîne de caractères, puis de réaffecter à cette même variable un nombre, sans avoir de message d'erreur. Voir l'exemple ci-dessous.

```
1 // JavaScript (non typé)
2 let stud1 = "Lola";
3 let note1 = 13;
4 //Réaffecté un nombre à la chaîne de caractère (String)
5 stud1 = 17;
6 //Réaffecté une chaîne de caractère au nombre
7 note1 = "Blabla";
8 console.log(stud1); // 17
```

[cf. ]

Certains développeurs diront que ceci est plutôt un avantage, mais cela peut causer beaucoup de confusion. Et pour être plus précis, la nature dynamique de JavaScript nous permet de faire virtuellement tout ce qu'on veut, et donc de faire toutes les erreurs possibles et imaginables aussi.

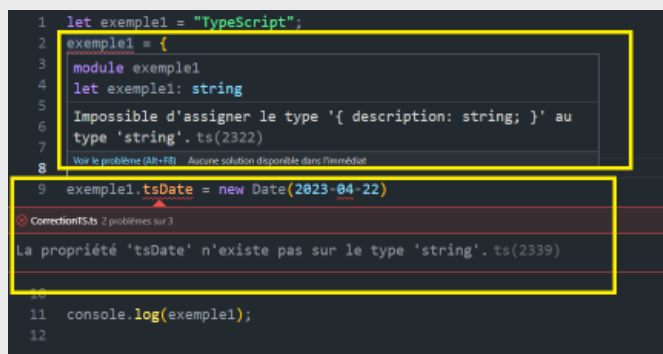
Si nous prenons un autre exemple, il y a au moins une erreur par ligne de code dans l'exemple ci-dessous, et les erreurs n'apparaissent qu'après exécution.

```
1 // Erreur de code JavaScript non détectable
2 let exemple1 = "TypeScript";
3 exemple1 = {
4     description : "Awesome TypeScript"
5 }
6 exemple1.tsDate = new Date(2023-04-22)
7
8 console.log(exemple1); //description: 'Awesome TypeScript', tsDate:...
9
```

[cf. ]

Si nous laissons ce code tel quel dans notre application, il risque de causer quantité de problèmes sans pouvoir déterminer la source des erreurs. En d'autres termes, notre application ne fonctionnera pas, et nous n'aurons pas la moindre idée de l'origine de l'erreur. Ce genre de problème peut faire perdre un temps considérable.

Nous parlons de JavaScript pour justement comprendre les avantages de TypeScript. TypeScript, par exemple, ne laissera pas passer ce genre d'erreurs. Comme nous le voyons sur la capture d'écran ci-dessous :



```
1 let exemple1 = "TypeScript";
2 exemple1 = {
3   module exemple1
4   let exemple1: string
5   Impossible d'assigner le type '{ description: string; }' au
6   type 'string'. ts(2322)
7   Voir le problème (Alt+F8) Aucune solution disponible dans l'inmédiate
8
9   exemple1.tsDate = new Date(2023-04-22)
10
11 console.log(exemple1);
12
```

Correction TS: 2 problèmes sur 3

La propriété 'tsDate' n'existe pas sur le type 'string'. ts(2339)

Comment ? En fait, TypeScript ajoute un typage statique fort, ce qui fait de lui un langage fortement typé, contrairement à JavaScript qui est un langage faiblement typé, voire non typé. Cela lui permet de vérifier le code au moment de la compilation et non pas à l'exécution.

## Les avantages et les inconvénients

Afin de mettre au clair les avantages ainsi que les inconvénients de TypeScript et JavaScript, voici une liste de leurs quelques atouts et limitations :

### Avantages :

TypeScript	JavaScript
<ul style="list-style-type: none"> <li>• Détection des erreurs de compilation : TypeScript est un langage typé, ce qui lui permet de détecter les erreurs avant l'exécution.</li> <li>• Il améliore la maintenabilité et la lisibilité du code (un atout majeur pour les grands projets).</li> <li>• Il intègre la programmation orientée objet : il prend en charge les concepts tels que les interfaces, l'héritage, les classes, etc.</li> <li>• Il peut être utilisé pour le côté serveur et le côté client.</li> <li>• Il prend en charge aussi bien le typage statique que le typage dynamique (avec type any).</li> <li>• Il peut utiliser les Bibliothèques de JavaScript.</li> <li>• Débogage plus facile.</li> <li>• Prise en charge par les IDE les plus populaires.</li> </ul>	<ul style="list-style-type: none"> <li>• Aucune installation n'est nécessaire : pas besoin d'installer quoi que ce soit, on peut coder et tester directement sur la console du navigateur.</li> <li>• Il convient aux apprentis (petites applications, facile à tester, etc.).</li> <li>• Il peut être utilisé pour le côté client et le côté serveur (avec node.js).</li> <li>• Facile à utiliser pour le côté client.</li> <li>• Exécution rapide (compilation non requise).</li> <li>• Offre une flexibilité avec le typage dynamique.</li> <li>• Syntaxe simple.</li> <li>• Fonctionne sur tous les navigateurs.</li> <li>• Il peut être utilisé pour le côté serveur et côté client.</li> </ul>

### Inconvénients :

TypeScript	JavaScript
<ul style="list-style-type: none"> <li>• Il nécessite d'être installé.</li> <li>• Les fichiers TypeScript(.ts) doivent être convertis (transpilés) en fichiers JavaScript(.js) avant exécution.</li> <li>• Il met plus de temps pour s'exécuter à cause de la compilation.</li> </ul>	<ul style="list-style-type: none"> <li>• Les erreurs ne sont détectées qu'après exécution.</li> <li>• Trop permissif : les erreurs ne sont pas détectées, même après exécution.</li> <li>• Ne convient qu'aux petits projets.</li> <li>• Difficile à déboguer.</li> <li>• Complexe pour le côté serveur.</li> </ul>

Notez bien que les inconvénients de TypeScript n'ont pas vraiment d'impact sur la qualité du code, nous avons juste une étape en plus à faire, qui est la **compilation**.

Cependant, si nous travaillons sur un autre *runtime* que Node.js, comme Deno ou Bun, par exemple, plus besoin de faire une compilation pour exécuter notre code ; ces derniers supportent nativement TypeScript. Plus loin dans le cours, nous allons voir ensemble les runtimes Deno et Node.js plus en détail.

De plus, beaucoup de frameworks *frontend* sont développés avec TypeScript comme Angular, Vue, Nest, etc. Nous avons aussi plusieurs runtimes qui ont émergé, qui support nativement TypeScript, tels que Bun, Deno, etc.

## B. Exercice : Quiz

[solution n°1 p.17]

### Question 1

Nous pouvons exécuter du code TypeScript sur n'importe quel navigateur.

- ☐ Vrai
- ☐ Faux

### Question 2

TypeScript est un langage multi-paradigme.

- ☐ Vrai
- ☐ Faux

### Question 3

Voici le code suivant :

```
1 let student1 = {
2     nom : "Toto",
3     note : 17
4 }
5 console.log(student1);
```

De quel langage de programmation s'agit-il ?

- ☐ JavaScript
- ☐ TypeScript
- ☐ TypeScript / JavaScript

### Question 4

TypeScript permet :

- ☐ D'améliorer la maintenabilité et la lisibilité du code
- ☐ De détecter les erreurs à la compilation
- ☐ D'intégrer la programmation orientée objet

### Question 5

JavaScript est un superset de TypeScript.

- ☐ Vrai
- ☐ Faux



### III. L'installation de TypeScript

#### A. Installation de TypeScript

##### Introduction

Dans cette partie, nous allons procéder à l'installation du compilateur TypeScript. Pas de contrainte au niveau du système d'exploitation, il est possible de l'utiliser sous Windows, Mac ou Linux.

L'installation n'est pas obligatoire si vous souhaitez travailler sur des runtimes (environnements d'exécutions) qui supportent nativement TypeScript, comme Deno ou Bun, ou directement sur le *Playground* de TypeScript `typescriptlang`<sup>1</sup>.

##### Comment exécuter du code JavaScript ?

Avant de voir comment installer et compiler TypeScript, nous allons d'abord voir comment exécuter du code JavaScript et où l'exécuter.

Le code JavaScript peut être exécuté dans différents types d'environnements. On peut l'exécuter soit dans un navigateur comme Chrome, Brave, le *playground* de TypeScript ou bien sur un runtime comme Node.js.

Méthode	Exécuter du code JavaScript sur un navigateur
	<p>En ce qui concerne le premier cas, c'est-à-dire les navigateurs, il suffit de lancer votre navigateur, et de faire un clic droit sur le navigateur, puis cliquer sur « <b>Inspecter</b> ». Vous pouvez également utiliser le raccourci clavier en appuyant sur la touche « <b>F12</b> ». Ces étapes sont valables pour tous les navigateurs.</p> <p>À première vue, nous avons beaucoup d'éléments - ce qui nous intéresse, c'est uniquement la console. Cliquez donc sur console. La console s'ouvre, nous pouvons tester l'exécution de notre code.</p> <pre>1 let student1 = { 2   nom : "Toto", 3   note : 17 4 } 5 console.log(student1);</pre> <p>[cf. ]</p> <p>Nous voyons bien qu'il nous renvoie notre valeur, donc, notre code fonctionne.</p>

Méthode	Exécuter du code JavaScript sur le Playground
	<p>Nous allons voir comment l'exécuter sur le <i>Playground</i> de TypeScript. Il suffit de cliquer sur le lien suivant : <code>typescriptlang</code><sup>2</sup> pour se rendre sur le Playground, puis d'insérer notre code et de cliquer sur « <b>run</b> ».</p> <p>Notez que le Playground est principalement utilisé pour compiler du code TypeScript en JavaScript, mais il sert aussi à exécuter du code JavaScript.</p>

Maintenant que nous avons vu comment exécuter notre code sur un navigateur et un playground, voyons comment l'exécuter sur un *runtime* ou un environnement d'exécution. Nous allons utiliser Node.js comme runtime.

Définition	Qu'est-ce que Node.js ?
	<p>Node.js est un runtime ou un environnement d'exécution JavaScript côté serveur, qui permet d'exécuter du code JavaScript en dehors d'un navigateur. Node.js nous permet de développer des applications côté serveur, des outils en ligne de commande, des scripts automatisés et bien plus encore.</p>

<sup>1</sup> <https://www.typescriptlang.org/play>

<sup>2</sup> <https://www.typescriptlang.org/play>

Il comprend une bibliothèque de modules intégrée, ce qui facilite le développement d'applications en offrant un accès à une grande variété de fonctionnalités. Il est souvent utilisé pour la création de serveurs web, mais il peut également être utilisé pour créer des applications de bureau, des applications mobiles, des jeux, etc.

#### Méthode Exécuter du code JavaScript sur un runtime

Pour pouvoir utiliser Node.js, nous allons devoir l'installer. Pour l'installer, il suffit de se rendre sur le site officiel de Node.js et de télécharger la version la plus récente, idéalement de télécharger la version stable, puis exécuter.

Mais avant, vérifions que Node.js s'est installé sur votre ordinateur en tapant la commande suivante : `node -v`. Cette commande nous donne aussi la version de Node.js installée. Une fois Node.js installé, nous pouvons procéder à l'exécution de notre code.

Prenons le même code que tout à l'heure pour le test :

```
1 let student1 = {
2   nom : "Toto",
3   note : 17
4 }
5 console.log(student1);
```

Si le code s'exécute, cela veut dire qu'il n'y a aucune erreur.

### Comment installer le compilateur TypeScript ?

Maintenant que nous avons vu comment exécuter du JavaScript, nous allons voir comment exécuter du TypeScript. Si nous essayons d'emblée de compiler le code TypeScript, cela ne fonctionnera pas, car nous devons d'abord installer un compilateur Typescript.

#### Définition Qu'est-ce qu'un compilateur TypeScript ?

Un compilateur TypeScript est un package NPM qui prend du code écrit en TypeScript et le traduit en code JavaScript (compréhensible par un navigateur web ou par un environnement d'exécution JavaScript comme Node.js). Il offre également des fonctionnalités avancées telles que la génération automatique de fichiers, la transformation de code basée sur des décorateurs, des options de configuration flexibles et bien plus encore.

#### Définition Qu'est-ce qu'un NPM ?

NPM signifie *Node Package Manager* ou gestionnaire de packages pour node, c'est une collection gigantesque de packages JavaScript (environ 2 millions de packages).

Il a au moins 17 millions d'utilisateurs à travers le monde, ce qui n'est pas négligeable. Son but, d'après ses fondateurs, est de rendre le développement avec JavaScript plus rapide et plus sûr.

#### Méthode Installer NPM

Voyons comment installer ce gestionnaire de packages NPM. Si nous nous rendons sur le site de NPM [npmjs Docs](https://docs.npmjs.com/downloading-and-installing-node-js-and-npm)<sup>1</sup>, il est écrit qu'il faut d'abord installer Node.js, et NPM est intégré lors de l'installation.

Comme nous avons déjà installé Node.js, NPM devrait également être installé. Node.js installe automatiquement NPM car il s'agit d'un package manager qui est souvent utilisé en conjonction avec Node.js pour installer des modules et des packages tiers. Pour vérifier si NPM s'est installé ou non, nous entrons la commande suivante dans un terminal : `npm -v`.

<sup>1</sup> <https://docs.npmjs.com/downloading-and-installing-node-js-and-npm>

Nous pouvons voir que NPM a bien été installé, car la version installée est affichée dans le terminal. Si NPM n'est pas installé, il est possible de l'installer en écrivant la commande suivante dans le terminal : `npm install -g npm`. Il est recommandé de le faire même s'il est déjà installé pour éviter de causer des bugs d'autorisation d'accès lors de l'utilisation des packages NPM.

Maintenant que nous avons installé NPM, nous pouvons désormais installer le package de compilation de TypeScript. Pour ce faire, il suffit d'exécuter la commande suivante : `npm install -g typescript` (-g signifie Global). Notez bien que nous avons installé TypeScript en global, mais nous pouvons très bien l'installer en local. Pour l'installer en local, il suffit de taper cette commande dans le terminal : `npm install typescript --save-dev`. Il est parfois préférable de l'installer en local lorsque nous avons plusieurs projets avec des versions différentes.

Pour ce qui nous concerne, nous avons plusieurs projets, mais avec les mêmes versions. De ce fait, il est plus simple d'installer en global, d'autant plus que nous passons d'un projet à un autre.

Pour tester si le compilateur TypeScript a bien été installé, nous allons exécuter la commande suivante : `tsc -v`, cela permet également de voir la version (-v pour version).

Maintenant que tout est bien installé, nous pouvons faire la compilation de notre code TypeScript. Pour ce faire, il suffit de taper notre code TypeScript dans l'IDE ou l'éditeur de texte, et de saisir cette commande : `tsc` sur le terminal, suivie du nom de fichier TypeScript que nous souhaitons compiler. Exemple : `tsc test1.ts` (le `.ts` n'est pas obligatoire)

```
1 //Objet TypeScript
2 let client = {
3     nom : "Aubergine",
4     tel : 777777777
5 }
6 console.log(client.nom);
```

#### Remarque

Nous sommes obligé d'apprendre à exécuter du code JavaScript pour coder avec Typescript, car nous devons compiler le code Typescript vers JavaScript pour pouvoir l'exécuter sur un navigateur, ou sur un environnement d'exécution comme Node.js. Cependant, si nous utilisons un environnement d'exécution qui supporte nativement TypeScript (comme Deno, Bun, etc.), il n'y a plus besoin de suivre ces étapes. De plus, si vous travaillez sur des frameworks frontend qui supportent nativement Typescript comme Angular, il n'y aura pas besoin de compiler.

## Compilation de projet TypeScript

Nous avons vu comment exécuter du code JavaScript, ensuite nous avons installé le compilateur TypeScript, puis nous avons compilé du code TypeScript. Nous allons maintenant compiler un projet TypeScript.

#### Méthode Comment compiler un projet TypeScript ?

Compiler un projet TypeScript en projet JavaScript, c'est comme compiler un fichier ou un code TypeScript en JavaScript, sauf que cette fois-ci, nous compilons plusieurs fichiers en même temps. Pour ce faire, il faut créer un fichier `tsconfig.json` qui permet de spécifier les différentes options pour dire au compilateur comment compiler le projet.

Voici un exemple de fichier `tsconfig.json` qui permet de spécifier les options de compilation :

```
1 {
2     "compilerOptions": {
3         "target": "ES6",                // Version de ECMAScript 6
4         "outDir": "dossierJS",         // Nom du dossier de distanciation pour les
5     },                                  fichiers JavaScript
6     "include": [
```

```
7      "**.ts"                                // Type de fichier à compiler
8    ]
9 }
```

Pour compiler le projet, il suffit de taper la commande `tsc` dans le terminal, le dossier contenant les fichiers JavaScript sera généré.

Notez bien qu'on a encore beaucoup d'options de configuration, mais ici, nous nous sommes limité au plus simple.

## B. Exercice : Quiz

[solution n°2 p.18]

### Question 1

Parmi cette liste, quel runtime environnement supporte nativement TypeScript ?

- ☐ Deno
- ☐ Nest
- ☐ Node.js

### Question 2

Quelle commande permet d'installer TypeScript en local ?

- ☐ `npm install typescript --save-dev`
- ☐ `npm install typescript`
- ☐ `npm install -g typescript`

### Question 3

Où peut-on exécuter du code JavaScript ?

- ☐ Navigateur
- ☐ Runtime
- ☐ Playground de TypeScript
- ☐ Toutes les réponses ci-dessus

### Question 4

Voici un extrait de code :

```
1 let greeting : string = "You look great";
```

Qu'allons-nous obtenir après compilation ?

- ☐ `let greeting = "You look great";`
- ☐ `let greeting : "You look great";`
- ☐ `let greeting = "You look great";`

### Question 5

Comment compiler un fichier TypeScript en fichier JavaScript ?

- ☐ En utilisant la commande `node` dans un terminal
- ☐ En utilisant la commande `tsc` dans un terminal
- ☐ En utilisant la commande `npm` dans un terminal

## IV. Essentiel

Plutôt que de créer un tout nouveau langage, TypeScript étend le langage JavaScript en mettant en avant ces avantages, tout en palliant ses imperfections. Ce qui fait de lui une sorte de version améliorée de JavaScript. Plus important encore, TypeScript donne une seconde vie à JavaScript.

En effet, TypeScript n'est pas le seul langage de script fortement typé. D'autres langages tels que CoffeeScript, Flow et Dart offrent également des fonctionnalités de typage statique pour les applications web.

Cependant, l'avantage de TypeScript n'est pas qu'une question de typage. Comme nous l'avons vu dans ce cours, l'un des avantages de TypeScript est qu'il est facile à apprendre pour les développeurs JavaScript. Étant donné que TypeScript est un surensemble de JavaScript, la syntaxe et les structures de JavaScript sont aussi valables en TypeScript. Cela signifie que les développeurs peuvent facilement passer de JavaScript à TypeScript et commencer à utiliser des fonctionnalités de typage statique sans avoir à apprendre une nouvelle syntaxe ou une nouvelle structure de langage.

En outre, TypeScript est largement soutenu par la communauté JavaScript et est utilisé dans de nombreux grands projets open source, ce qui signifie que les développeurs peuvent bénéficier de la richesse des ressources et des outils disponibles pour TypeScript.

Pour finir, l'avantage de TypeScript réside dans sa combinaison de fonctionnalités de typage statique et de compatibilité avec JavaScript, ce qui le rend facile à apprendre et à utiliser pour les développeurs JavaScript.

## V. Auto-évaluation

### A. Exercice

#### Question 1

[solution n°3 p.19]

1. Vérifiez que le compilateur TypeScript a bien été installé sur votre machine.
2. Créez un nouveau fichier TypeScript avec le contenu suivant :

```
1 function bonjour(name: string) {  
2   console.log(`Bonjour, ${name} !`);  
3 }  
4 bonjour("Votre nom");
```

3. Compilez le fichier TypeScript en fichier JavaScript, puis exécutez-le.

#### Question 2

[solution n°4 p.19]

Veuillez créer un projet TypeScript, puis créer votre fichier `tsconfig.json` (vous pouvez utiliser la commande `tsc --init` pour générer le fichier `tsconfig.json`). Le but est que lorsque vous exécutez le compilateur TypeScript, il doit générer des fichiers JavaScript dans un dossier cible. Pour cet exercice, utilisez la fonction `function somme (num1: number, num2: number)` qui permet d'additionner 2 chiffres et la fonction `function hello ()` qui affichera Bonjour et votre nom.

Voici des exemples de code :

```
1 // Fonction pour faire la somme  
2 function somme(num1: number, num2: number) {  
3   console.log(num1 + num2);  
4 }  
5 somme(15, 10);
```

```
1 // Fonction de salutation
2 function hello(prenom: string) {
3     console.log(`Bonjour ${prenom}`);
4 }
5 hello("Messy");
```

## B. Test

### Exercice 1 : Quiz

[solution n°5 p.20]

#### Question 1

NPM veut dire :

- ☐ Node Package Manager
- ☐ Gestionnaire de paquets pour Node.js
- ☐ Collection de paquets JavaScript
- ☐ Toutes les réponses ci-dessus

#### Question 2

Voici un extrait de code :

```
1 let x = "Chaine de caractères"
2 x = 15
```

Ce code est-il valide en JavaScript et TypeScript ?

- ☐ Valide uniquement sur TypeScript
- ☐ Valide uniquement sur JavaScript
- ☐ Valide sur JavaScript et TypeScript

#### Question 3

Quel outil supporte nativement TypeScript ?

- ☐ Deno
- ☐ Bun
- ☐ Les deux

#### Question 4

Qui a créé et développé TypeScript ?

- ☐ Apple
- ☐ Oracle
- ☐ Microsoft
- ☐ Google

#### Question 5

Parmi la liste ci-dessous, quels sont les avantages de TypeScript ?

- ☐ Facilite la structuration du code
- ☐ Impose des directives
- ☐ Utilise la programmation récursive

## Solutions des exercices






**Exercice p. 8 Solution n°1****Question 1**

Nous pouvons exécuter du code TypeScript sur n'importe quel navigateur.

☐ Vrai

☒ Faux

 TypeScript ne peut être exécuté sur un navigateur, il faut le compiler en JavaScript.

**Question 2**

TypeScript est un langage multi-paradigme.

☒ Vrai

☐ Faux

 TypeScript est un langage Programmation Fonctionnelle et de Programmation Orientée Objet.

**Question 3**

Voici le code suivant :

```
1 let student1 = {  
2     nom : "Toto",  
3     note : 17  
4 }  
5 console.log(student1);
```

De quel langage de programmation s'agit-il ?

☐ JavaScript

☐ TypeScript

☒ TypeScript / JavaScript

 La déclaration d'objets en JavaScript et en TypeScript est identique.


**Question 4**

TypeScript permet :

☒ D'améliorer la maintenabilité et la lisibilité du code


☒ De détecter les erreurs à la compilation

☒ D'intégrer la programmation orientée objet

 TypeScript présente de nombreux avantages comme d'améliorer la maintenabilité et la lisibilité du code, d'intégrer la programmation orientée objet, etc.

**Question 5**


JavaScript est un superset de TypeScript.

- ☐ Vrai
- ☒ Faux
-  TypeScript est un superset (sur-ensemble) de JavaScript, ce qui veut dire que JavaScript est un sous-ensemble de TypeScript.

## Exercice p. 12 Solution n°2


### Question 1

Parmi cette liste, quel runtime environnement supporte nativement TypeScript ?

- ☒ Deno
- ☐ Nest
- ☐ Node.js
-  Deno est un environnement d'exécution (runtime) qui supporte nativement TypeScript, Deno n'a pas besoin d'un compilateur externe pour exécuter du code TypeScript. Il peut exécuter directement du code TypeScript sans nécessiter de compilation préalable.


### Question 2

Quelle commande permet d'installer TypeScript en local ?

- ☒ `npm install typescript --save-dev`
- ☐ `npm install typescript`
- ☐ `npm install -g typescript`
-  La commande `npm install typescript --save-dev` permet d'installer Typescript en local. Il est préférable de l'installer en local lorsque vous avez plusieurs projets avec des versions différentes.

### Question 3

Où peut-on exécuter du code JavaScript ?


- ☐ Navigateur
- ☐ Runtime
- ☐ Playground de TypeScript
- ☒ Toutes les réponses ci-dessus
-  Nous pouvons exécuter le code JavaScript sur la console du navigateur, un runtime ou le playground de TypeScript.

### Question 4

Voici un extrait de code :


```
1 let greeting : string = "You look great";
```

Qu'allons-nous obtenir après compilation ?

- ☒ `let greeting = "You look great";`
- ☐ `let greeting : "You look great";`
- ☐ `let greeting = "You look great";`
-  JavaScript est du TypeScript sans le typage, donc, il suffit de supprimer le typage `string`.

### Question 5

Comment compiler un fichier TypeScript en fichier JavaScript ?

- ☐ En utilisant la commande `node` dans un terminal
- ☒ En utilisant la commande `tsc` dans un terminal
- ☐ En utilisant la commande `npm` dans un terminal
-  La commande `tsc` est une ligne de commande qui permet d'exécuter le programme *TypeScript Compiler*, elle permet donc de transformer le code écrit en TypeScript en JavaScript.

#### p. 13 Solution n°3

1. Ouvrez votre terminal ou votre invite de commande, et tapez la commande suivante : `tsc --version` ou `tsc --v`.
2. Vous pouvez utiliser le code dans l'énoncé pour créer votre fichier.
3. Utilisez la commande `tsc index.ts` dans votre terminal.
4. Après compilation, vous allez obtenir un code comme ci-dessous :

```
1 // Code après compilation
2 function bonjour(name) {
3     console.log(`Bonjour, ${name} !`);
4 }
5 bonjour("Votre nom"); // Bonjour Votre nom
```

[cf.]

#### p. 13 Solution n°4

Après compilation, nous devons avoir un dossier avec les fichiers JavaScript, qui devront pouvoir s'exécuter.

Voici un exemple de code après compilation :

```
1 // Fonction pour faire la somme après compilation
2 function somme(num1, num2) {
3     console.log(num1 + num2);
4 }
5 somme(15, 10);


1 // Fonction de salutation après compilation
2 function hello(prenom) {
3     console.log(`Bonjour ${prenom}`);
4 }
5 hello("Messy");
```

[cf.]

## Exercice p. 14 Solution n°5

### Question 1

NPM veut dire :


- ☐ Node Package Manager
- ☐ Gestionnaire de paquets pour Node.js
- ☐ Collection de paquets JavaScript
- ☒ Toutes les réponses ci-dessus
-  NPM est Node Package Manager ou gestionnaire de paquets pour Node.js, c'est une collection gigantesque de packages JavaScript.

### Question 2

Voici un extrait de code :


```
1 let x = "Chaîne de caractères"
2 x = 15
```

Ce code est-il valide en JavaScript et TypeScript ?

- ☐ Valide uniquement sur TypeScript
- ☒ Valide uniquement sur JavaScript
- ☐ Valide sur JavaScript et TypeScript
-  JavaScript est typé dynamiquement, donc nous pouvons assigner à la même variable des valeurs de types différents.


### Question 3

Quel outil supporte nativement TypeScript ?

- ☐ Deno
- ☐ Bun
- ☒ Les deux
-  Bun et Deno sont des *runtime* qui supporte nativement TypeScript.

### Question 4


Qui a créé et développé TypeScript ?

- ☐ Apple
- ☐ Oracle
- ☒ Microsoft
- ☐ Google
-  TypeScript a été développé par Microsoft, le principal auteur est Anders Hejlsberg le cocréateur de C#.

**Question 5**

---

Parmi la liste ci-dessous, quels sont les avantages de TypeScript ?

- ☒ Facilite la structuration du code
- ☒ Impose des directives
- ☐ Utilise la programmation récursive
-  TypeScript facilite la structuration du code et impose des directives. Cependant, il est utilisé pour de la programmation orientée objet.