TypeScript : introduction

Achref El Mouelhi

Docteur de l'université d'Aix-Marseille
Chercheur en programmation par contrainte (IA)
Ingénieur en génie logiciel

elmouelhi.achref@gmail.com

Plan

TypeScript

ECMA

- Pour **E**uropean **C**omputer **M**anufacturers **A**ssociation.
- Créé en 1961.
- Devenu **Ecma International** - European association for standardizing information and communication systems en 1994.
- Organisme de standardisation pour plusieurs domaines de l'informatique
  - les langages de script
  - les produits de sécurité
  - la structure de fichier
  - ...

TypeScript

ECMAScript

- Ensemble de normes sur les langages de programmation de type script
  - **JavaScript**
  - **VBScript**
  - **AppleScript**
  - ...
- Standardisé par **Ecma International** depuis 1994

TypeScript

Quelques versions

- Version 5 (**ES5**) ou **ES2009**
- Version 6 (**ES6**) ou **ES2015** (compatible avec les navigateurs modernes)
- Version 7 appelé **ES2016** (ne s'appelle plus ES7)
- Version 8 appelé **ES2017**
- Version 9 appelé **ES2018**
- Version 10 appelé **ES2019**
- Version 11 appelé **ES2020**
- Version 12 appelé **ES2021**

TypeScript

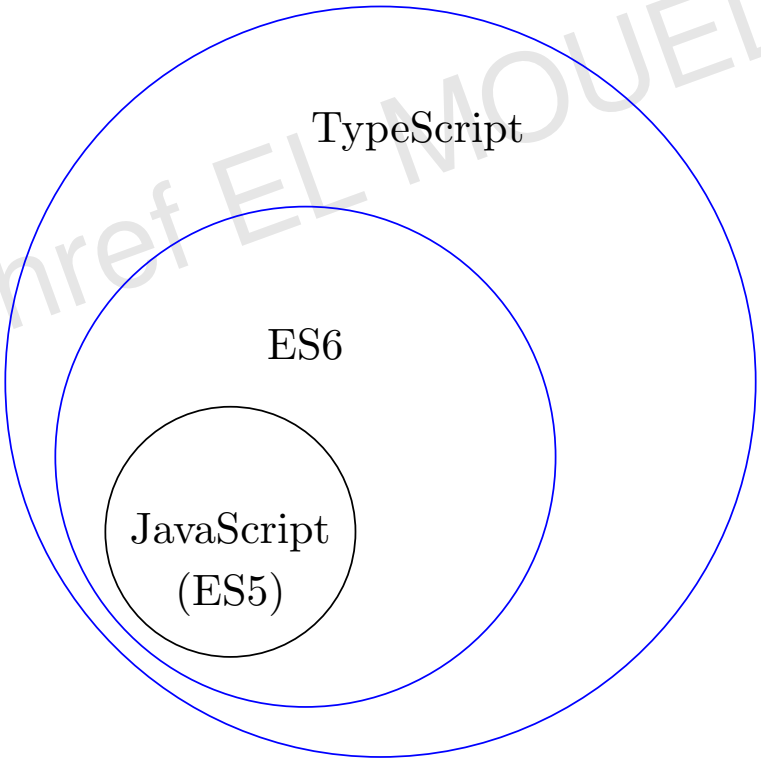
TypeScript

- langage de programmation
  - procédural et orienté-objet
  - supportant le typage statique, dynamique et générique
- open-source
- créé par Anders Hejlsberg (Créateur de **C#**) de **MicroSoft**

TypeScript

TypeScript : initialement une sur-couche de **ES6** ajoutant

- typage statique
- meilleure gestion de module (à ne pas confondre avec les modules **Angular**)



TypeScript

TypeScript

- Sur-couche de **ES6**.
- Supportant plusieurs concepts introduits dans des versions plus récentes (> ES6).
- Pouvant être configuré pour supporter des versions plus récentes.

TypeScript

Frameworks Frontend utilisant **TypeScript**

- **Angular** depuis la version 2
- **React** (voir `https://www.typescriptlang.org/docs/handbook/react.html`) et **Vue.js** (voir `https://fr.vuejs.org/v2/guide/typescript.html`) qui intègrent un support **TypeScript**

TypeScript

Frameworks Frontend utilisant **TypeScript**

- **Angular** depuis la version 2
- **React** (voir `https://www.typescriptlang.org/docs/handbook/react.html`) et **Vue.js** (voir `https://fr.vuejs.org/v2/guide/typescript.html`) qui intègrent un support **TypeScript**

Frameworks Backend utilisant **TypeScript**

- **ExpressJS** avec l'extension **TypeScript Node Starter** (voir `https://github.com/microsoft/TypeScript-Node-Starter`)
- **nest** Framework backend de **TypeScript**

TypeScript

Librairie écrite en **TypeScript**

RxJS (voir `https://github.com/ReactiveX/rxjs`)

TypeScript

Librairie écrite en **TypeScript**

RxJS (voir `https://github.com/ReactiveX/rxjs`)

API écrite en **TypeScript**

Firebase (voir `https://github.com/firebase/`)

TypeScript

Librairie écrite en **TypeScript**

RxJS (voir `https://github.com/ReactiveX/rxjs`)

API écrite en **TypeScript**

Firebase (voir `https://github.com/firebase/`)

Deno

Le nouveau projet de Ryan Dahl pour remplacer **NodeJS** écrit en **TypeScript**

TypeScript

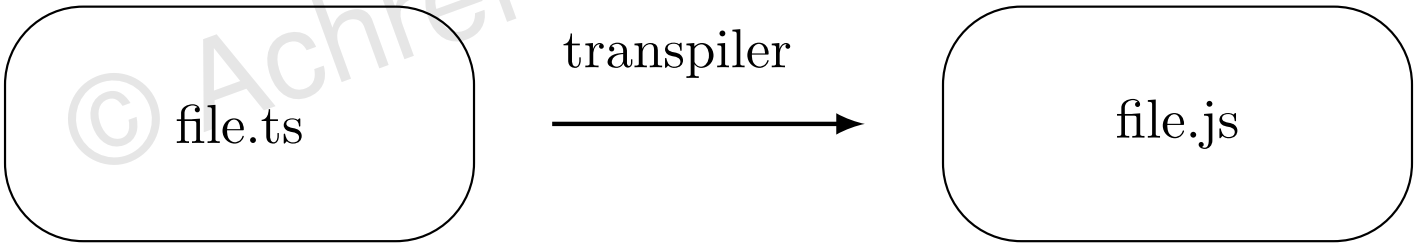
Le navigateur ne comprend pas **TypeScript**

Il faut le transcompiler (ou transpiler) en **JavaScript**

TypeScript

Le navigateur ne comprend pas **TypeScript**

Il faut le transcompiler (ou transpiler) en **JavaScript**

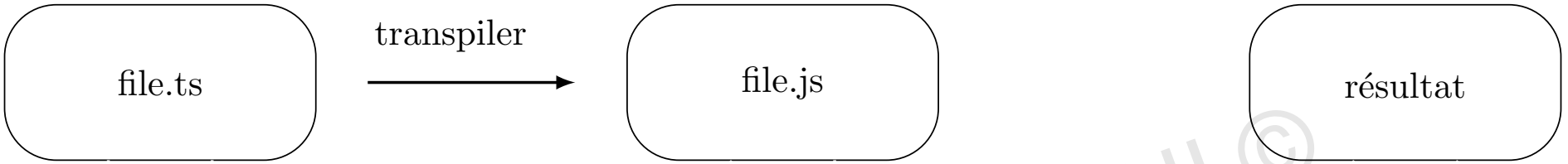


TypeScript

Comment va t-on procéder dans ce cours ?

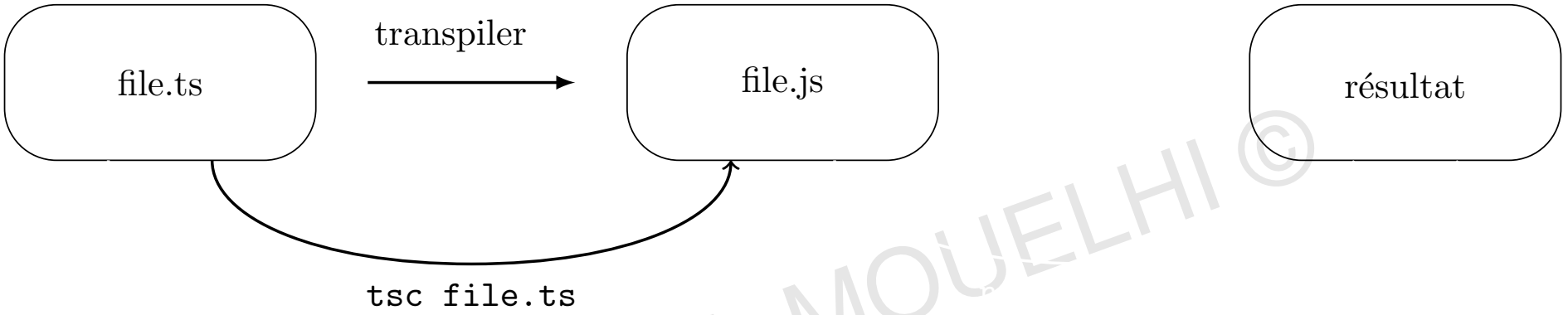
TypeScript

Comment va t-on procéder dans ce cours ?



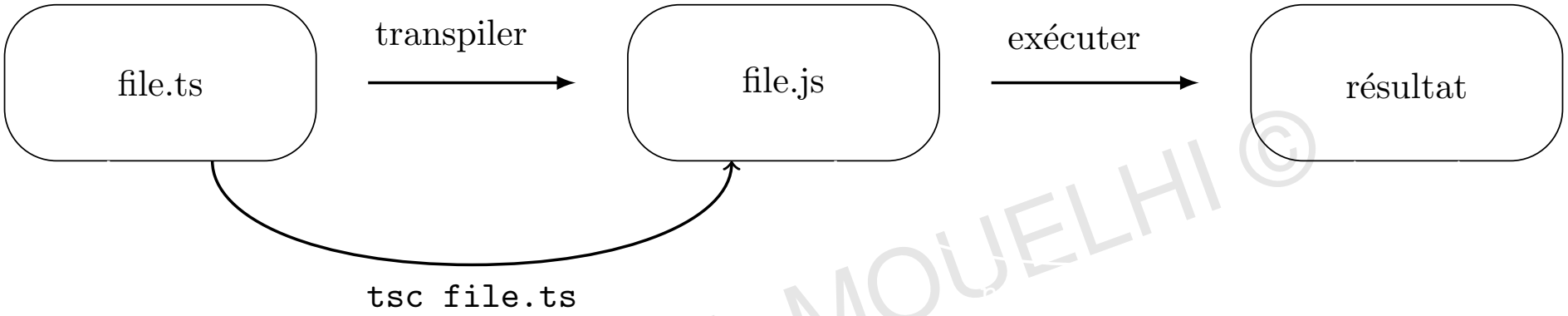
TypeScript

Comment va t-on procéder dans ce cours ?



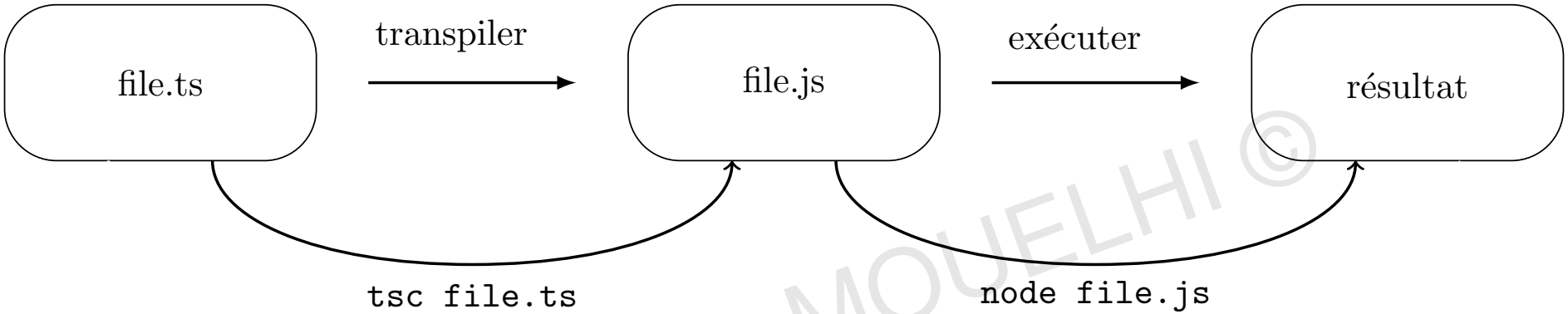
TypeScript

Comment va t-on procéder dans ce cours ?



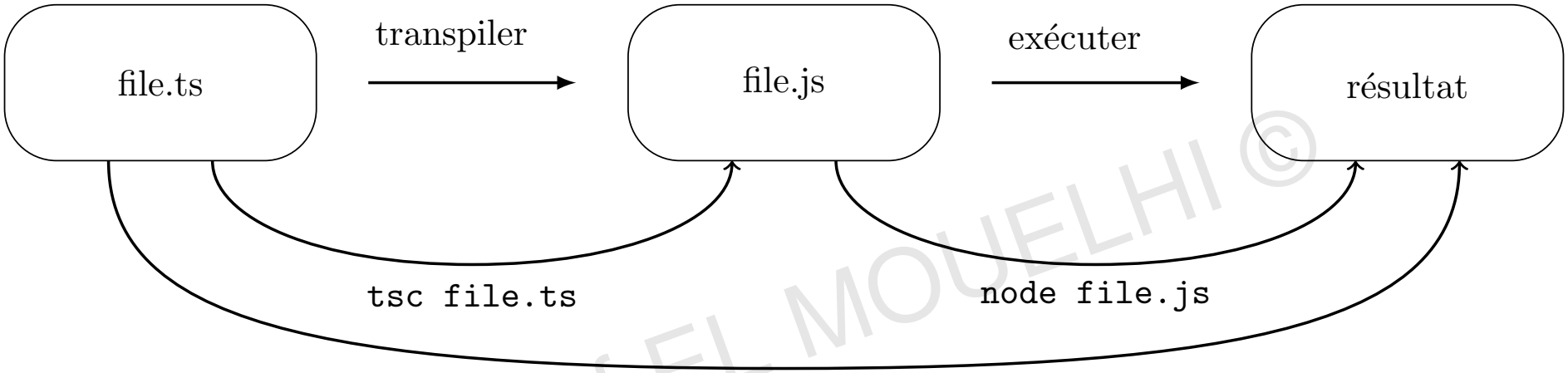
TypeScript

Comment va t-on procéder dans ce cours ?



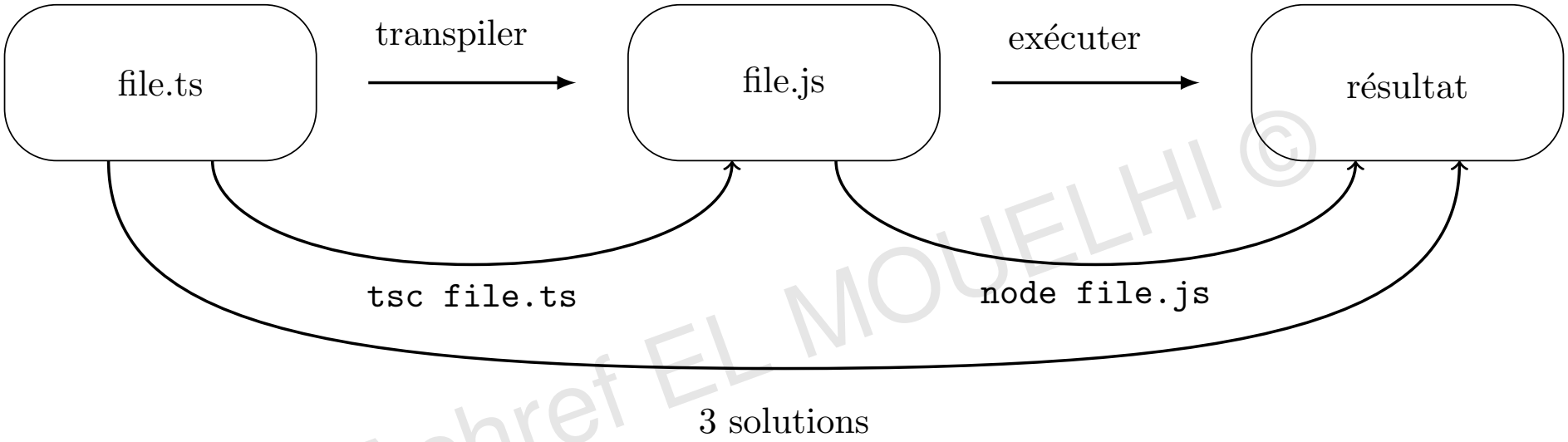
TypeScript

Comment va t-on procéder dans ce cours ?



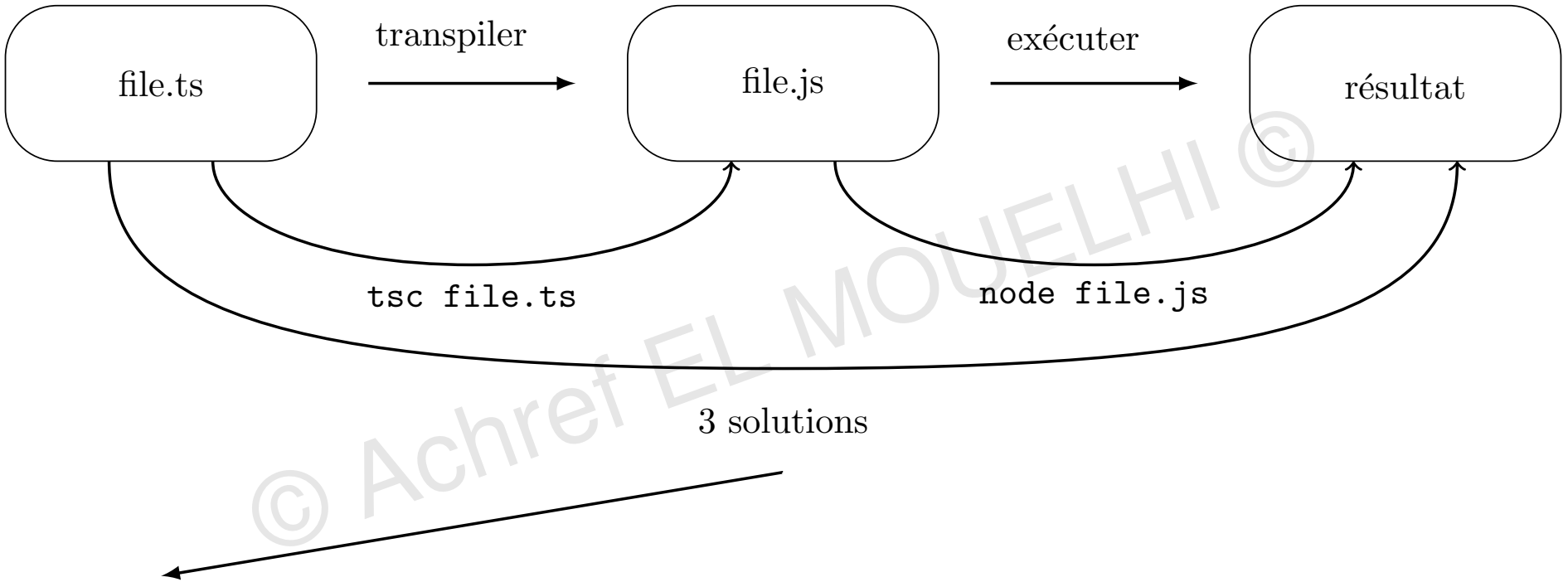
TypeScript

Comment va t-on procéder dans ce cours ?



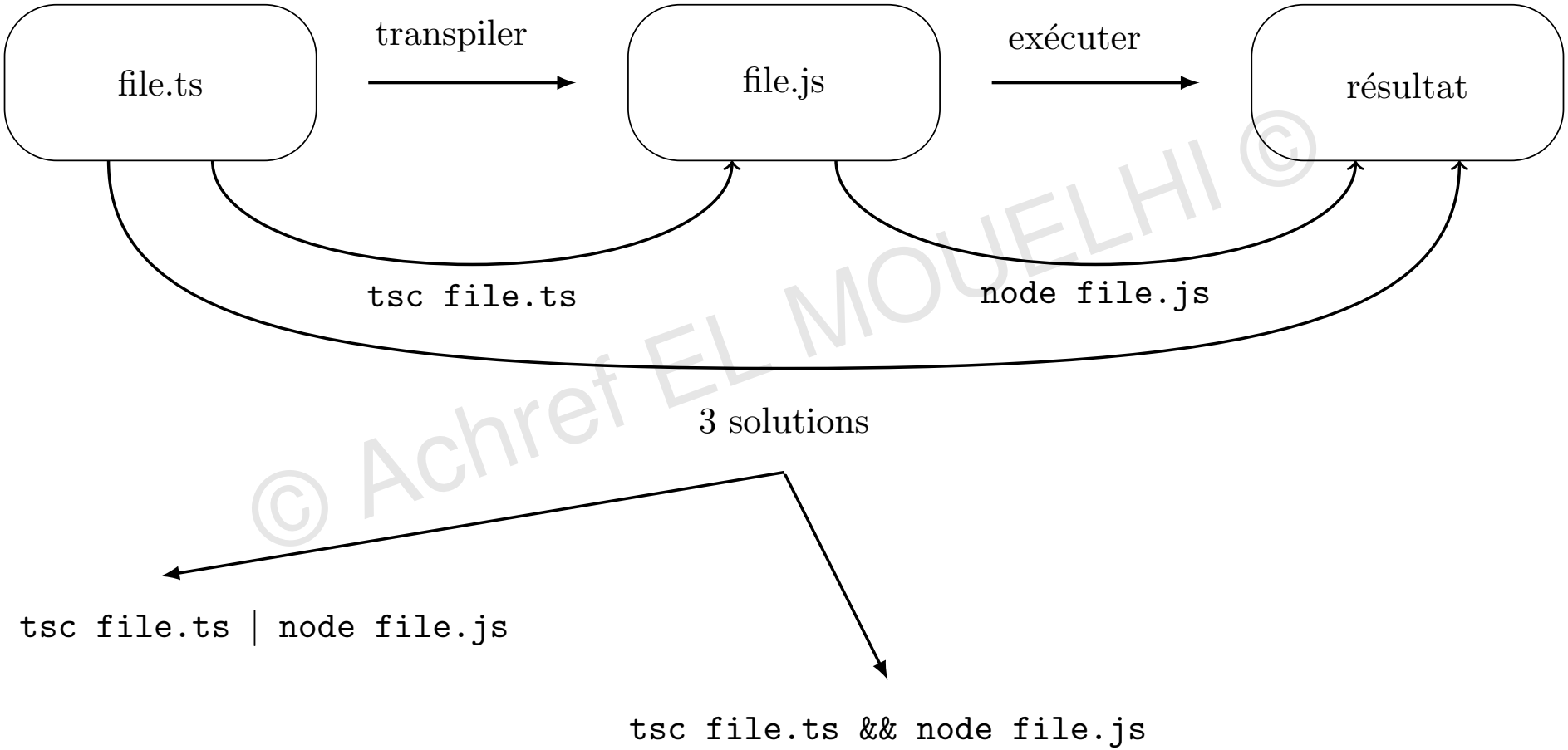
TypeScript

Comment va t-on procéder dans ce cours ?



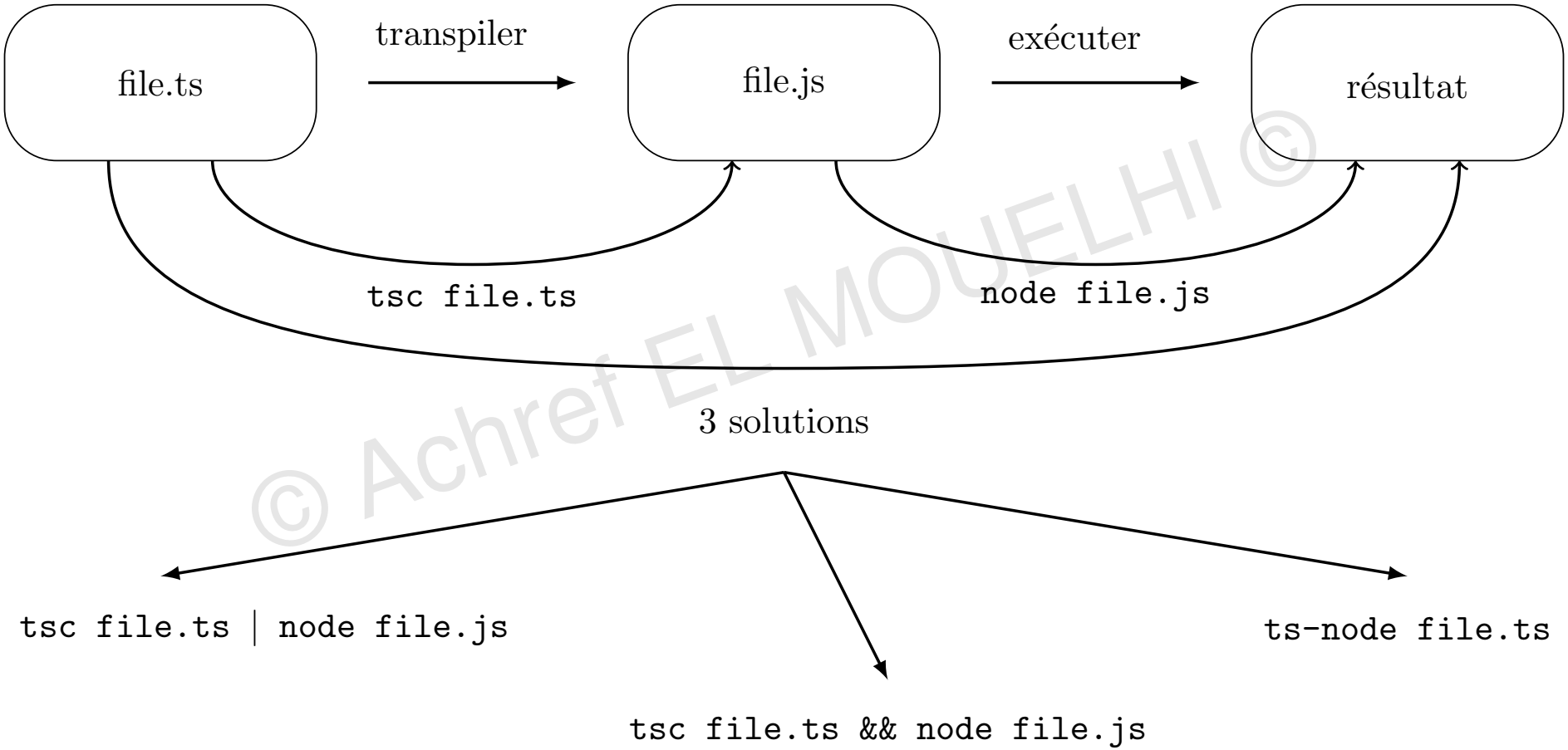
TypeScript

Comment va t-on procéder dans ce cours ?



TypeScript

Comment va t-on procéder dans ce cours ?



TypeScript

De quoi on a besoin ?

- **Node.js** pour exécuter la commande `node`
- **TypeScript** pour exécuter la commande `tsc`
- **ts-node** pour exécuter la commande `ts-node`

TypeScript

Pour **Node.js**, il faut

- aller sur `https://nodejs.org/en/`
- choisir la dernière version, télécharger et installer

TypeScript

Pour **Node.js**, il faut

- aller sur `https://nodejs.org/en/`
- choisir la dernière version, télécharger et installer

Pour vérifier l'installation depuis une console (invite de commandes), exécutez

```
node -v
```

TypeScript

Pour installer TypeScript, ouvrez une console (invite de commandes) et exécutez

```
npm install -g typescript
```

TypeScript

Pour installer TypeScript, ouvrez une console (invite de commandes) et exécutez

```
npm install -g typescript
```


Pour vérifier l'installation depuis une console (invite de commandes), exécutez

```
tsc -v
```

TypeScript

Pour consulter la liste d'options pour la commande `tsc`

`https://www.typescriptlang.org/docs/handbook/compiler-options.html`

TypeScript

Pour consulter la liste d'options pour la commande `tsc`

`https://www.typescriptlang.org/docs/handbook/compiler-options.html`

Ou, exécutez depuis une console (invite de commandes)

```
tsc --help
```

TypeScript

Pour installer ts-node, ouvrez une console (invite de commandes) et exécutez

```
npm install -g ts-node
```

TypeScript

Pour installer ts-node, ouvrez une console (invite de commandes) et exécutez

```
npm install -g ts-node
```

Pour vérifier l'installation depuis une console (invite de commandes), exécutez

```
ts-node -v
```

TypeScript

Les règles de nommage

- Pour classes et interfaces : **Pascal case**
- Pour variables, fonctions et méthodes : **Camel case**
- Pour fichiers : **Kebab case**

TypeScript

Les règles de nommage

- Pour classes et interfaces : **Pascal case**
- Pour variables, fonctions et méthodes : **Camel case**
- Pour fichiers : **Kebab case**

Pour plus de détails

`https://wprock.fr/blog/conventions-nommage-programmation/`

TypeScript

Quel IDE (Environnement de développement intégré) pour **TypeScript** ?

- **Visual Studio Code** (À ne pas confondre avec Visual Studio)
- Visual Studio
- Eclipse
- ...

TypeScript

Visual Studio Code (ou VSC) , pourquoi ?

- Gratuit.
- Offrant la possibilité d'intégrer des éditeurs de texte connus (comme **Sublime Text**, **Atom**...).
- Extensible selon le langage de programmation.
- Recommandé par **Microsoft** (créateur de **TypeScript**) pour les projets **Angular**.

TypeScript

Visual Studio Code (ou VSC) , pourquoi ?

- Gratuit.
- Offrant la possibilité d'intégrer des éditeurs de texte connus (comme **Sublime Text**, **Atom**...).
- Extensible selon le langage de programmation.
- Recommandé par **Microsoft** (créateur de **TypeScript**) pour les projets **Angular**.

VSC : téléchargement

`code.visualstudio.com/download`

TypeScript

Quelques raccourcis **VSC**

- Pour activer la sauvegarde automatique : aller dans `File > AutoSave`
- Pour indenter son code : `Alt` `Shift` `f`
- Pour commenter/décommenter : `Ctrl` `:`
- Pour faire une sélection multiple : `Ctrl` `f2`
- Pour sélectionner l'occurrence suivante : `Ctrl` `d`

TypeScript

Démarche

- Créez un répertoire `cours-ts` dans votre espace de travail
- Lancez VSC et allez dans `File > Open Folder...` et choisissez `cours-ts`
- Dans `cours-ts`, créez un fichier `file.ts`

TypeScript

Fichier de configuration : `tsconfig.json`

- Situé à la racine du projet **TypeScript**
- Consulté par le compilateur à chaque exécution de la commande `tsc`
- Si le fichier n'existe pas, des valeurs par défaut seront utilisées (`https://www.typescriptlang.org/docs/handbook/compiler-options.html`)

TypeScript

Fichier de configuration : `tsconfig.json`

- Situé à la racine du projet **TypeScript**
- Consulté par le compilateur à chaque exécution de la commande `tsc`
- Si le fichier n'existe pas, des valeurs par défaut seront utilisées (`https://www.typescriptlang.org/docs/handbook/compiler-options.html`)

Pour le générer, exécutez

```
tsc --init
```

TypeScript

Contenu de `tsconfig.json` généré (sans les commentaires)

```
{
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true
  }
}
```

TypeScript

Ajoutons l'instruction suivante dans `file.ts`

```
console.log(2 ** 3);
```

TypeScript

Ajoutons l'instruction suivante dans `file.ts`

```
console.log(2 ** 3);
```

Lancez la commande

```
tsc
```

TypeScript

Ajoutons l'instruction suivante dans `file.ts`

```
console.log(2 ** 3);
```

Lancez la commande

```
tsc
```

Vérifions le contenu suivant dans `file.js`

```
"use strict";
console.log(Math.pow(2, 3));
```

TypeScript

Explication

- L'opérateur `**` a été introduit dans ES2016.
- La valeur de la clé `target` dans `tsconfig.json` indique que le code **TypseCript** sera transpilé en ES5.
- `"use strict"` a été ajouté car la clé `strict` a la valeur `true` dans `tsconfig.json`.

TypeScript

Modifions la valeur de la propriété `target` dans `tsconfig.json`

```
"target": "es2016"
```

TypeScript

Modifions la valeur de la propriété `target` dans `tsconfig.json`

```
"target": "es2016"
```

Relancez la commande

```
tsc
```

TypeScript

Modifions la valeur de la propriété `target` dans `tsconfig.json`

```
"target": "es2016"
```

Relancez la commande

```
tsc
```

Vérifions le contenu suivant dans `file.js`

```
"use strict";
console.log(2 ** 3);
```

TypeScript

Lancez la commande `tsc` en spécifiant le nom du fichier

```
tsc file.ts
```

TypeScript

Lancez la commande `tsc` en spécifiant le nom du fichier

```
tsc file.ts
```

Vérifions le contenu suivant dans `file.js`

```
console.log(Math.pow(2, 3));
```

TypeScript

Lancez la commande `tsc` en spécifiant le nom du fichier

```
tsc file.ts
```

Vérifions le contenu suivant dans `file.js`

```
console.log(Math.pow(2, 3));
```


Remarque

Si on précise un nom de fichier après la commande `tsc`, le fichier `tsconfig.json` sera ignoré.

TypeScript

Pour transpiler le code TypeScript en ES2016, exécutez

```
tsc file.ts --target es2016
```

TypeScript

Pour transpiler le code TypeScript en ES2016, exécutez

```
tsc file.ts --target es2016
```

Ou

```
tsc file.ts -t es2016
```

TypeScript

Créons un deuxième fichier `file2.ts` avec le contenu suivant

```
console.log("Hello World!");
```

TypeScript

Créons un deuxième fichier `file2.ts` avec le contenu suivant

```
console.log("Hello World!");
```

Lancez la commande

```
tsc
```

TypeScript

Créons un deuxième fichier `file2.ts` avec le contenu suivant

```
console.log("Hello World!");
```

Lancez la commande

```
tsc
```


Remarque

Vérifier que les deux fichiers `file.ts` et `file2.ts` ont été transpilé en `file.js` et `file2.js`.

Pour ne pas transpiler `file2.ts`, ajoutons la clé `exclude` dans `tsconfig.json`

```
{
  "exclude": ["file2.ts"],
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true
  }
}
```

Pour ne pas transpiler `file2.ts`, ajoutons la clé `exclude` dans `tsconfig.json`

```
{
  "exclude": ["file2.ts"],
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true
  }
}
```

Supprimez les fichiers `.js` avant de lancer

```
tsc
```

Pour ne pas transpiler `file2.ts`, ajoutons la clé `exclude` dans `tsconfig.json`

```
{
  "exclude": ["file2.ts"],
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true
  }
}
```

Supprimez les fichiers `.js` avant de lancer

```
tsc
```

Remarque

Vérifier que seul le fichier `file.ts` a été transpilé en `file.js`.

Pour transpiler seulement `file2.ts`, ajoutons la clé `include` dans `tsconfig.json`

```
{
  "include": ["file2.ts"],
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true
  }
}
```

Pour transpiler seulement `file2.ts`, ajoutons la clé `include` dans `tsconfig.json`

```
{
  "include": ["file2.ts"],
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true
  }
}
```

Supprimez les fichiers `.js` avant de lancer

```
tsc
```

Pour transpiler seulement `file2.ts`, ajoutons la clé `include` dans `tsconfig.json`

```
{
  "include": ["file2.ts"],
  "compilerOptions": {
    "target": "es5",
    "module": "commonjs",
    "esModuleInterop": true,
    "forceConsistentCasingInFileNames": true,
    "strict": true,
    "skipLibCheck": true
  }
}
```

Supprimez les fichiers `.js` avant de lancer

```
tsc
```

Remarque

Vérifier que seul le fichier `file2.ts` a été transpilé en `file2.js`.