Note

Nous allons faire une transformation d'un n'importe langage sous forme de BNF à un langage intermédiaire TypeScript.

D'un fichier TypeScript, on aura besoin un outil PlantUML pour obtenir un diagramme UML.

Etape 1

En utilisant Langium, définir un langage.

Yo langium

Extention…

Fichier de la grammaire : src/language/LanguageName.langium

Etape 2

Modifier language/generated/validator.ts

Import model,fonction from ast.js

Tester langage : mode debugger

Etape 3

Parser : de ts to js to json, breakpoint

Fichier d’origine : ast.ts

Commande: node -e "require('./test.cjs')('../src/language/generated/ast.ts', {})"

Obtenir comme:

{

Assignement: { '$container': 'Model', left: 'VarRef', value: 'number' },

Model: { statements: 'Array' },

VarDecl: { '$container': 'Model', name: 'string' },

VarRef: { '$container': 'Assignement', ref: 'Reference' }

}

Debug

npm run doIt

detail:

re structurer : yo node

update module : npm install --global generator-node

Reference

<https://stackoverflow.com/questions/39588436/how-to-parse-typescript-definition-to-json/45832978#45832978>

Fichier parser.js from M. Deantoni

Etape 4

Semaine 1

Connaitre langium, générer un langage personnalisé. « définir «  la grammaire, analyse ast.ts fichier, modifier validator.ts => éditeur de la langage

<https://github.com/microsoft/TypeScript/wiki/Using-the-Compiler-API>

Semaine 2

Exécuter parser.js au mode debugge, faire un stop point pour savoir où il se parcours dans le AST

Connaitre «visite » ast automatiquement

1. Construire un Parser de TypeScript

2) Génération de parser

Outils de génération de parsers : "TypeScript Compiler" ou "tsc."

Le TypeScript Compiler est le composant central du projet TypeScript qui prend en charge l'analyse du code TypeScript, la génération d'AST (Abstract Syntax Trees), la vérification du type, la compilation en JavaScript et d'autres fonctionnalités liées à TypeScript.

Le TypeScript Compiler est plus qu'un simple parser ; il s'agit d'un ensemble complet d'outils pour la gestion de code TypeScript, y compris la gestion de modules, la vérification de type, la génération de déclarations de types, etc. Pour la plupart des projets, il n'est pas nécessaire de créer un parser TypeScript personnalisé, car le TypeScript Compiler fournit déjà toutes les fonctionnalités nécessaires pour travailler avec du code TypeScript.

Pour l’utiliser :

-Tsc hello.ts //utilise TypeScript Compiler pour convertir un fichier.ts en un fichier JavaScript.

-node hello.js //exécuter le code JavaScript généré

3) Définir la grammaire TypeScript

Cette grammaire doit prendre en compte tous les éléments du langage, y compris les expressions, les déclarations, les types, etc.

4) Générer le code du parser

Générer le code du parser en utilisant la grammaire de PlantUML

5) Intégrer le parser dans votre projet

Intégrez le code généré du parser dans votre projet. Cela peut nécessiter quelques ajustements pour qu'il s'intègre correctement avec le reste de votre application.

6) Analyser du code TypeScript

Parcourir l'arbre syntaxique résultant pour effectuer différentes tâches, telles que la validation, la transformation ou la génération de documentation.