Making PT accessible

Implementing PT in TypeScript

Petter Sæther Moen

Thesis submitted for the degree of
Master in Informatics: Programming and System Architecture
60 credits

Department of Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Autumn 2020

Making PT accessible

Implementing PT in TypeScript

Petter Sæther Moen

© 2020 Petter Sæther Moen

Making PT accessible

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Acknowledgements

Contents

List of Figures

List of Tables

Preface

Part I

Introduction

Chapter 1

Introduction

1.1 What is PT?

1.2 Purpose of implementing PT in TS

Chapter 2

Background

2.1 Package Templates

2.2 TypeScript

Part II

The project

Chapter 3

Planning the project

3.1 Why TypeScript

3.2 Choosing the right approach

Before jumping into a project of this size it is important to find out what approach to use. The end goal of this project is to extend TypeScript with the Package Templates language mechanism, this can be achieved as following:

- Making a fork of the TypeScript compiler
- Making a preprocessor for the TypeScript compiler
- Making a Babel plugin
- Making a TypeScript compiler plugin / transformer

To get a better understanding for which approach to use we will be implementing a small language feature with each of the approaches named above.

3.2.1 TypeScript Compiler Fork

Possible, however not as accessible as other alternatives and will make upkeep expensive.

3.2.2 Preprocessor for the TypeScript Compiler

A lot more work than ex plugin / transformer.

3.2.3 TypeScript Compiler Plugin / Transform

As of the time of writing this the official TypeScript compiler does not support compile time plugins, the plugins for the TypeScript compiler is, as the TypeScript compiler wiki specifies, "for changing the editing experience only"[3]. However there are alternatives that do enable compile time plugins / transformers;

- ts-loader[5], for the webpack ecosystem
- Awesome Typescript Loader[4], for the webpack ecosystem. Deprecated
- ts-node[6], REPL / runtime
- ttypescript[2], TypeScript tool TODO: Les mer på dette
- A compiler wrapper

3.2.4 Babel plugin

Babel isn't strictly for TypeScript, but for JavaScript as a whole. Will however make it very accessible as most web-projects use Babel, and the upkeep is cheap, as plugins are pretty independent from the core.

Saying that this is strictly a Babel plugin wouldn't be entirely true, as we would have to fork the Babel parser in order to include our custom syntax[1]. However this is all hidden away from the user, as this custom parser is a dependency of our Babel plugin.

Babel Plugin is not as nice as I first thought, it seems to be pretty hard to write third-party plugins as you have to make a parser fork for custom syntax, and there is a severe lack of documentation. Most examples of Babel plugins mostly use internal helpers and utils, which are hard to use for third-party plugins.

TODO: Does it support having multiple custom parser? E.g. babel-plugin-typescript + our custom babel plugin?

Part III

Conclusion

Chapter 4

Results

Bibliography

[1] Babel. *@babel/parser*. URL: https://babeljs.io/docs/en/babel-parser (visited on 25/09/2020).

[2] cevek. *ttypescript*. URL: https://github.com/cevek/ttypescript.

[3] Microsoft. *microsoft/TypeScript*. URL: https://github.com/microsoft/TypeScript/wiki/Writing-a-Language-Service-Plugin.

[4] Stanislav Panferov. *Awesome TypeScript Loader*. URL: https://github.com/s-panferov/awesome-typescript-loader.

[5] TypeStrong. *ts-loader*. URL: https://github.com/TypeStrong/ts-loader.

[6] TypeStrong. *ts-node*. URL: https://github.com/TypeStrong/ts-node.