TYPO3 Documentation

Release 0.1.0

Elmar Hinz

June 07, 2016

Contents

CHAPTER 1

Introduction

This extensions ships a TypoScript parser, that is suited to replace the original TypoScript parser for frontend rendering.

1.1 What it is not

1.1.1 No Boost in Performance

The parser is approximately twice as fast as the origional parser, but there will be no boost of performance, as TypoScript parsing takes just a few milliseconds at all. You will not feel a difference.

1.1.2 Not Used in the Backend

The parser still doesn't suppurt syntax highlighting and error handling. For now it is not used in the backend at all.

The backend features will follow in future versions.

1.2 What it is

1.2.1 New Architecture

The reason to write a new TypoScript parser is, to get a modern architecture for it. The architecture is clean and easy to understand. The final goal is, to get a parser that makes it easy to develop TypoScript into the future.

- easy to understand
- easy to debug
- easy to extend

1.2.2 Standalone Usage

It's possible to use the TypoScript parser standalone outside of the TYPO3 CMS if you like the TypoScript syntax an want to use it for configuration in other fields.

1.2.3 Condition Preprocessor

Condition evaluation is done by a preprocessor. By separtion of the condition preprocessing it becomes possible to use the TypoScript parser without bothering with conditions.

On the other hand by isolating the conditions it becomes possible to enhance the conditions easily. Nested conditions would be an example of enhancement.

As with the old parser the condition matching is handled by a third object. This enables the development of conditions, that address a completly different field than the TYPO3 CMS.

1.2.4 Public Presentation

This is a public presentation of the parser. Should it replace the old parser of the core? If yes, it needs to be tested in the wild before until it is really stable. This is the extension to do so.

1.3 Differences

- Backslash doesn't escape anything.
- Escaping of dots in object keys is not supported.
- Backslash is an allowed character in the keys (for PHP namespaces).

CHAPTER 2

Administration

Install the extension, clear caches and check of your frontend is rendered as expected.

If things go wrong, uninstall.

2.1 Technical Implementation

The origional parser is not fully replaced but extended by XCLASS registration. This registration is only done in FE mode. The extended class serves as adapter to the standalone classes.

CHAPTER 3

Lessons Learned

The overall time to parse the TypoScript of a website takes just a few milliseconds. It is not a critical part of the overall page rendering time. Yet the development of this extension was also focused on performance.

3.1 Time to parse the templates vs. time to parse TypoScript

When measured with the TYPO3 core time tracker (admin panel) the template parsing takes a few hundred milliseconds. When measuring and summing up all calls to the TypoScript parse function (TypoScriptParser::parse()) it takes just a few milliseconds. The difference is most likley to be explained by I/O calls to read the templates.

3.2 Non-Recursive Parser

The `Non-Recursive Parser` is the approach taken by this parser. The whole rendering happens within one function by using simple loop structures. Calls to itself or other methods are avoided as far as reasonable. This turns out to be twice as fast as the recursive `Original TypoScript Parser`.

3.3 Original TypoScript Parser

The original parser of the TYPO3 core uses recursive calls to handle the nesting of the braces of the object name pathes.

3.4 JSON Parser

The idea of the `JSON Parser` was, to use the PHP function `json_decode` to create the large `TypoScript` tree consisting of hundreds of PHP arrays on the binary level. `TypoScript` was rewritten to a valid `JSON` string as input.

Unfortunately `json_decode` does merging but not recursive merging. As overwriting is a feature of `TypoScript` this requires to prepare the `JSON` rendering by any approach to do the overwriting in advance. An array was created, containing the full object path as key and the value as value to solve this. Although this creates no nested tree, it takes time.

Together with the conversion to a `JSON` string in the second step, there is no advantage in speed. Taking the non-recursive approach to handle the two steps, it ends up in a similar speed as the `Original TypoScript Parser`.