

Method

This chapter contains a detailed guide through the various steps and components of the Nickel Language Server (NLS). Being written in the same language (Rust[@rust]) as the Nickel interpreter allows NLS to integrate existing components for language analysis. Complementary, NLS is tightly coupled to Nickel's Syntax definition. Hence, in sec. ?? this chapter will first detail parts of the AST that are of particular interest for the LSP and require special handling. Based on that sec. ?? will introduce the main datastructure underlying all higher level LSP interactions and how the AST is transformed into this form. Finally, in sec. ?? the implementation of current LSP features is discussed on the basis of the previously reviewed components.

Nickel AST

Nickel's Syntax tree is a single sum type, i.e. an enumeration of node types. Each enumeration variant may refer to child nodes, representing a branch or hold terminal values in which case it is considered a leaf of the tree. Additionally, nodes are parsed and represented, wrapped in another structure that encodes the span of the node and all its potential children.

Basic Elements

The data types of the Nickel language are closely related to JSON. On the leaf level, Nickel defines `Boolean`, `Number`, `String` and `Null` types. In addition to that the language implements native support for `Enum` values.

Completing JSON compatibility, `List` and `Record` types are present as well. Records on a syntax level are HashMaps, uniquely associating an identifier with a sub-node.

These data types constitute a static subset of Nickel which allows writing JSON compatible expressions as shown in lst. 0.1.

Listing 0.1 Example of a static Nickel expression

```
{  
  list = [ 1, "string", null ],  
  "enum value" = `Value  
}
```

Meta Information

Records

Static access

Linearization

States

Distinguished Elements

Transfer from AST

Retrying

Post-Processing

Resolving Elements

LSP Server

Diagnostics and Caching

Capabilities

Hover

Completion

Jump to Definition

Show references