

Template

v0.0.1

<https://github.com/SillyFreak/typst-my-package>

Clemens Koza

ABSTRACT

A template for typst packages

CONTENTS

I Introduction	2
II Module reference	3

I INTRODUCTION

This is a template for typst packages. It provides the `parse()` and `eval()` functions:

```
lib.typ
1 #let parse(expr) = { typ
2   // this is the "interesting" part: calling into the Rust parser
3   cbor.decode(_p.parse(cbor.encode(expr)))
4 }
5 /// -> int
6 #let eval(expr, ..vars) = {
7   assert(vars.pos().len() == 0)
8   let vars = vars.named()
9
10  let inner-eval(expr) = {
11    if expr.type == "number" { expr.value }
12    else if expr.type == "variable" { vars.at(expr.name) }
13    else if expr.type == "binary" {
14      let (operator, left, right) = expr
15      (left, right) = (inner-eval(left), inner-eval(right))
16      if operator == "add" { left + right }
17      else if operator == "sub" { left - right }
18      else if operator == "mul" { left * right }
19      else if operator == "div" { left / right }
20      else { panic("unexpected binary operator: " + operator) }
21    }
22    else { panic("unexpected expression type: " + expr.type) }
23  }
24
25  inner-eval(parse(expr))
26 }
```

Here they are in action:

```
1 $2 * (2 + x) arrow.double.long$ #my-package.parse("2 * (2 + x)") typ
```

$$2 * (2 + x) \Rightarrow ($$

type: "binary",

operator: "mul",

left: (type: "number", value: 2),

right: (

type: "binary",

operator: "add",

left: (type: "number", value: 2),

right: (type: "variable", name: "x"),

),

)

```
1 $2 * (2 + x) arrow.double.long^(x=3)$ #my-package.eval("2 * (2 + x)", x: 3) typ
```

$$2 * (2 + x) \stackrel{x=3}{\Rightarrow} 10$$

II MODULE REFERENCE

II.a my-package

- `parse()`

- `eval()`

```
parse(expr: str) -> dict
```

Parses an expression via a WASM plugin.

```
1 #my-package.parse("1") \
2 #my-package.parse("1 + 1") \
3 #my-package.parse("foo") \
4 #my-package.parse("foo + 1")
```

typ

```
(type: "number", value: 1)
(
  type: "binary",
  operator: "add",
  left: (type: "number", value: 1),
  right: (type: "number", value: 1),
)
(type: "variable", name: "foo")
(
  type: "binary",
  operator: "add",
  left: (type: "variable", name: "foo"),
  right: (type: "number", value: 1),
)
```

Parameters:

`expr (str)` – the expression to parse

```
eval(expr: str, ..vars: arguments) -> int
```

Evaluates an expression in Typst, by traversing the abstract syntax tree (AST) created in Rust.

```
1 #my-package.eval("1") \
2 #my-package.eval("1 + 1") \
3 #my-package.eval("foo", foo: 1) \
4 #my-package.eval("foo + 1", foo: 1)
```

typ

```
1
2
1
2
```

Parameters:

`expr (str)` – the expression to evaluate

`..vars (arguments)` – the variable assignments in the expression