PUC HIGHLIGHTING (VS-CODE EXTENSION)

Fabian Jülich
Tim Köhne

AGENDA

Preview

Syntax und Semantik

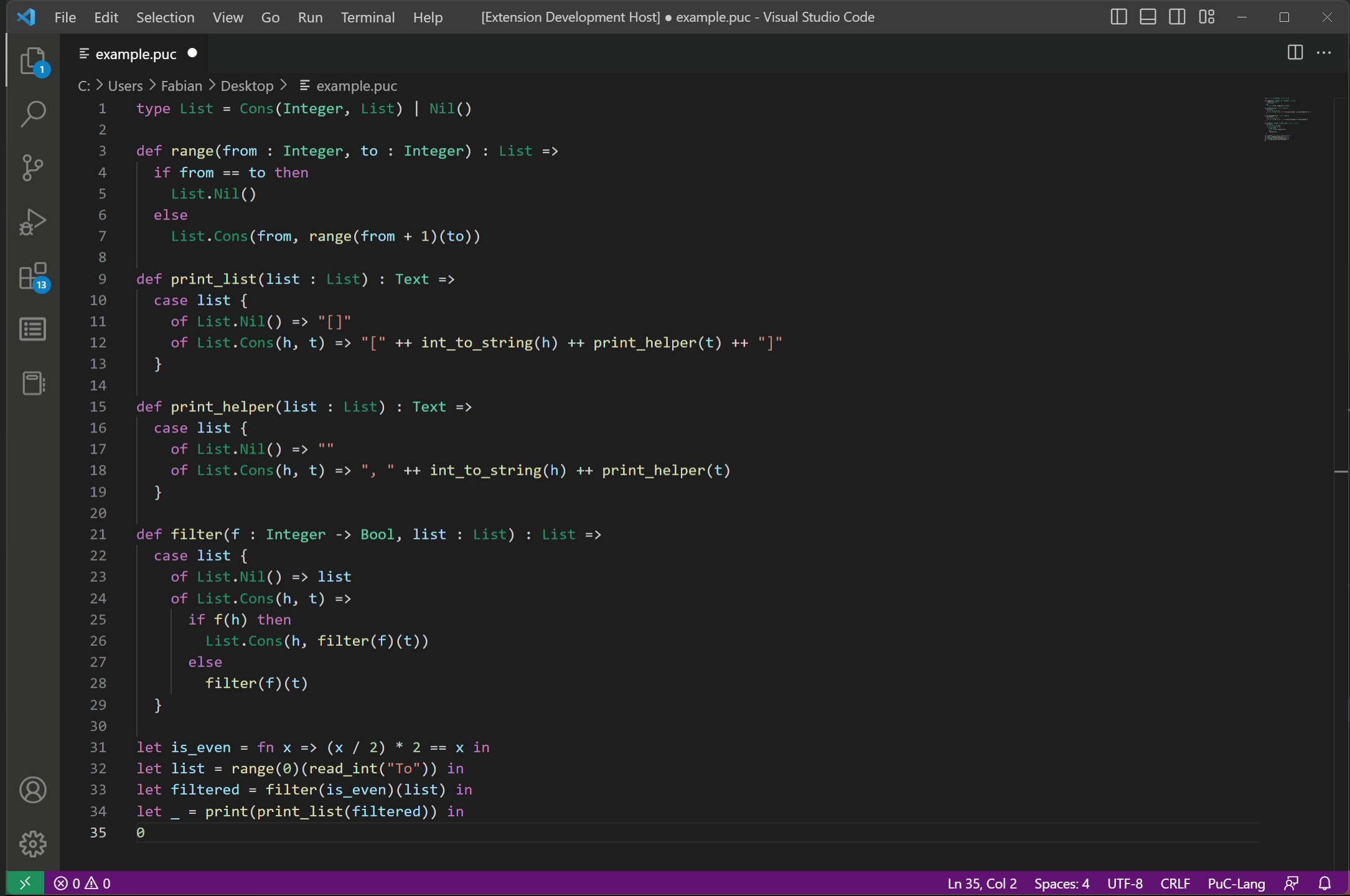
VS-Code Extensions

Umsetzung

Ausblick

PREVIEW

Vorstellung der Funktionsweise


File Edit Selection View Go Run Terminal Help
[Extension Development Host] ● example.puc - Visual Studio Code
example.puc
C: > Users > Fabian > Desktop > example.puc
type List = Cons(Integer, List) | Nil()
def range(from : Integer, to : Integer) : List =>
if from == to then
List.Nil()
else
List.Cons(from, range(from + 1)(to))
def print_list(list : List) : Text =>
case list {
of List.Nil() => "[]"
of List.Cons(h, t) => "[" ++ int_to_string(h) ++ print_helper(t) ++ "]"
}
def print_helper(list : List) : Text =>
case list {
of List.Nil() => ""
of List.Cons(h, t) => ", " ++ int_to_string(h) ++ print_helper(t)
}
def filter(f : Integer -> Bool, list : List) : List =>
case list {
of List.Nil() => list
of List.Cons(h, t) =>
if f(h) then
List.Cons(h, filter(f)(t))
else
filter(f)(t)
}
let is_even = fn x => (x / 2) * 2 == x in
let list = range(0)(read_int("To")) in
let filtered = filter(is_even)(list) in
let _ = print(print_list(filtered)) in
0
⊗ 0 ⚠ 0
Ln 35, Col 2
Spaces: 4
UTF-8
CRLF
PuC-Lang

SYNTAX UND SEMANTIK

Begriffserklärung

Syntax

- Strukturelle Regeln und Konventionen
- Beschreibt erlaubte Ausdrücke und Konstrukte
- Basierend auf mathematischer Notation (Lambda-Kalkül)
- Funktionen, Argumente, Musterabgleich, Rekursion
- Ausdrucksorientierte Auswertung
- Klar, präzise und lesbar (wichtig für Fehlerminimierung und Wartbarkeit)

Semantik

- Bedeutung und Interpretation von Programmen
- Beschreibt, wie Programme funktionieren und was sie tun
- Definiert das Verhalten von Ausdrücken und Konstrukten
- Bezieht sich auf die Ausführung und Auswertung von Code
- Regelt den Effekt von Funktionen und Operationen auf Daten
- Umfasst Typsysteme und Regeln für Fehlerbehandlung
- Konsistente und vorhersagbare Programmausführung

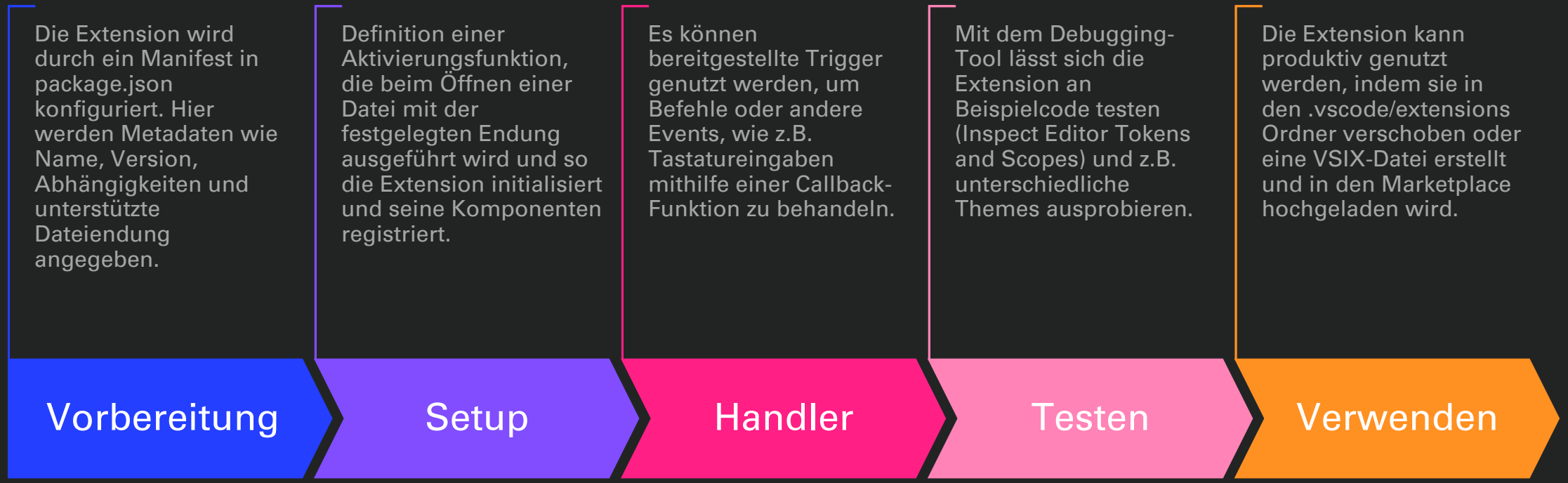
VS-CODE EXTENSIONS

Einführung

Abhängigkeiten

- Microsoft Visual Studio Code (Code Editor)
- npm (Paketmanager für JS-Bibliotheken
- TypeScript Compiler (kompiliert zu JavaScript)
- Node.js (Laufzeitumgebung für JS)
- Optional: VS Code Extension Development Kit (Tool zum Erstellen und Verwalten von Extensions, welches das Packen und Veröffentlichen vereinfacht)

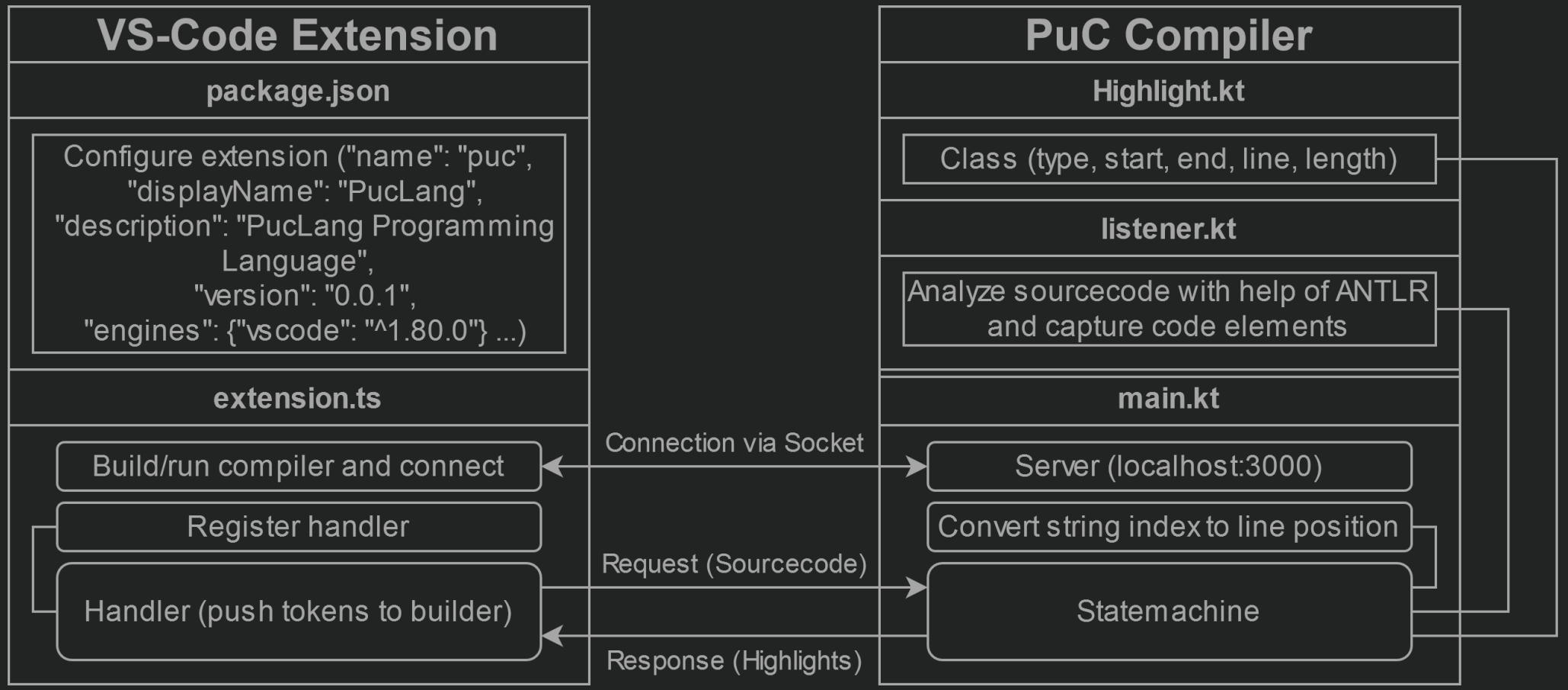
Extensions entwickeln



UMSETZUNG

Spezifische Implementierung

Architektur



AUSBLICK

Erweiterbarkeit

Wie könnte es weitergehen?

Die Extension ist beliebig erweiterbar und lässt somit zu, auch das Kompilieren und Ausführen von PuC Quellcode oder Features wie Autovervollständigung zu implementieren, um so eine vollwertige Sprachunterstützung zu schaffen.

VIELEN DANK!

https://github.com/timkoehne/Semantic-Highlighting-for-PuC-SS23