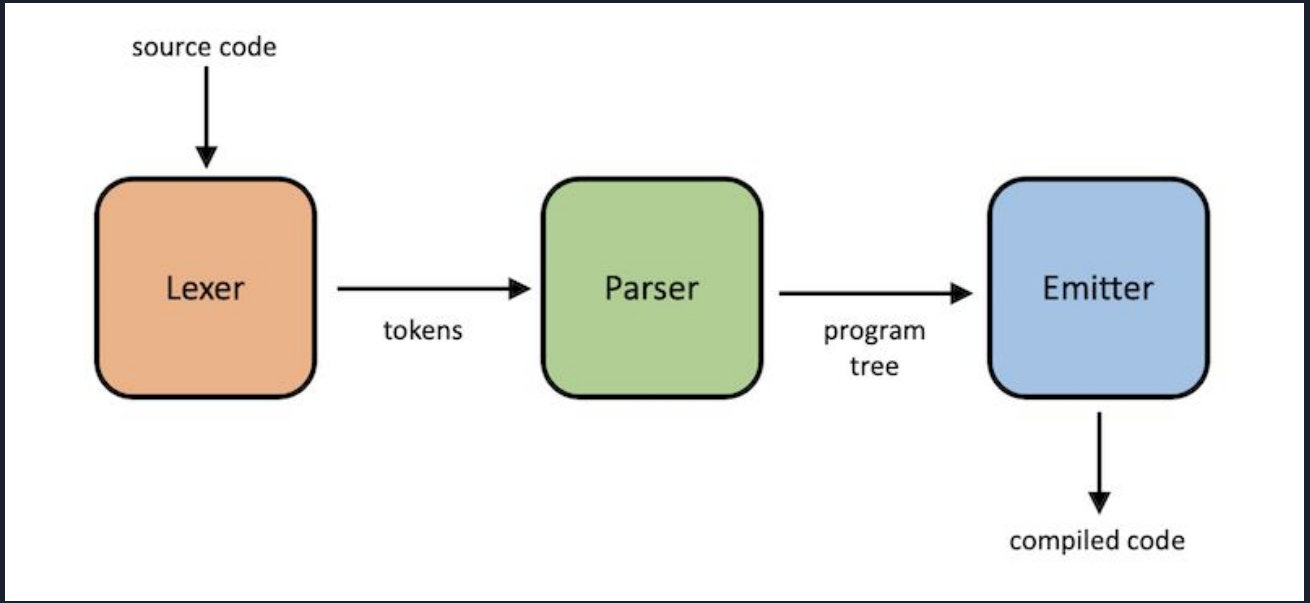
# Transpiler naar javascript

Kobe De Broeck - Liam Leirs - Jens Vissenberg - Stein Vandenbroeke

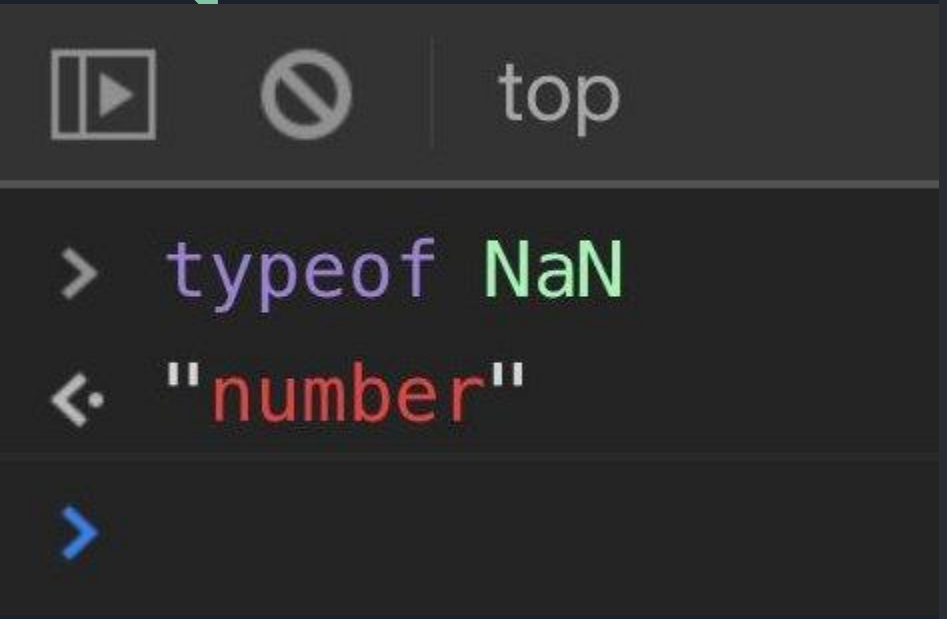
# Wat gaan we doen



# Doel:

- fouten vermijden
- minder debuggen
- minder javascript gebruiken

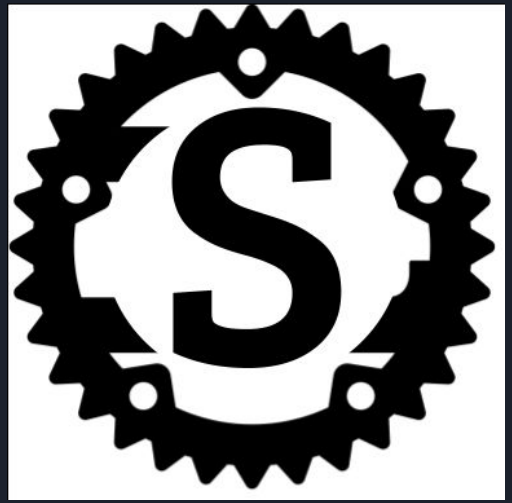
# Javascript




```
console.log(0 == 0)
  //true
console.log(0 == '0')
  //true
console.log(0 === 0)
  //true
console.log(0 === '0')
  //false
```

# STaal (Solide Taal)

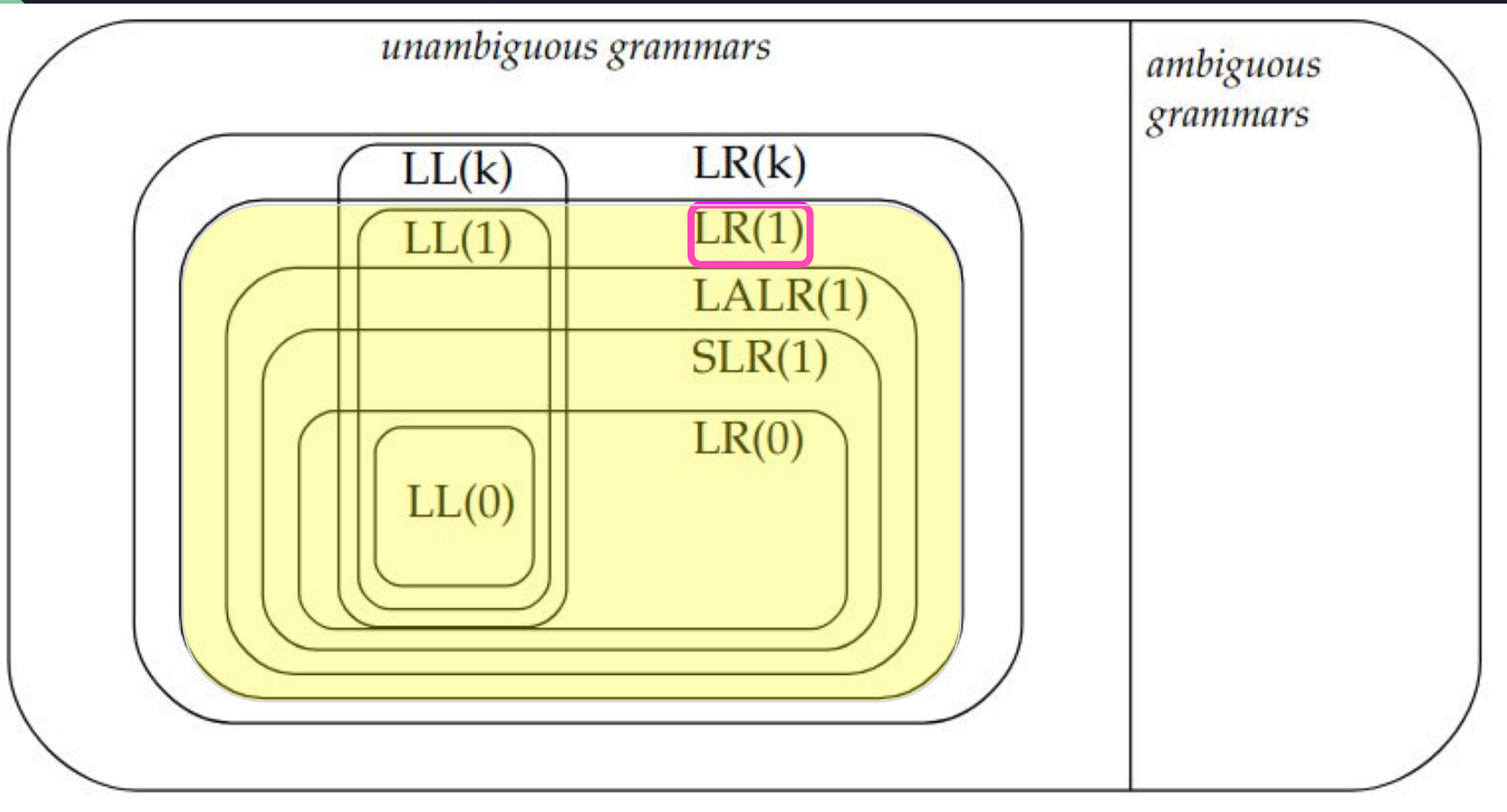
- variable initialisatie/declaratie
- loop
- optelling, aftrekking, vermenigvuldigen, deling
- if, else statement
- print



# CFG voor taal

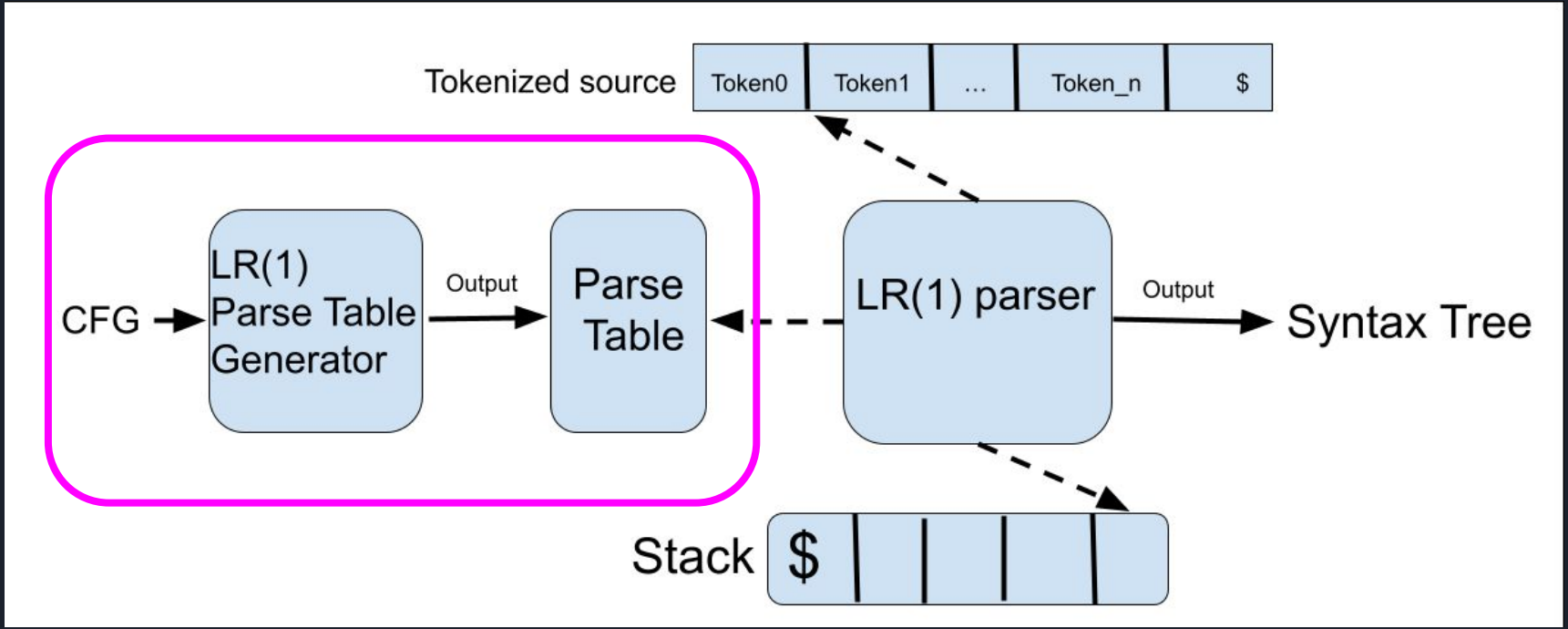
# Tokenizer m.b.v. $\varepsilon$NFA

- Tekens naar tokens
- Tokens houden hun posities in de source code bij (lijnnummer + positie in die lijn)
  - Nuttig voor bv.
    - foutmeldingen
    - synchronisatie tussen tree-view en source-view
- while string => token ***while***


unambiguous grammars
ambiguous grammars
LL(k)
LR(k)
LL(1)
LR(1)
LALR(1)
SLR(1)
LR(0)
LL(0)

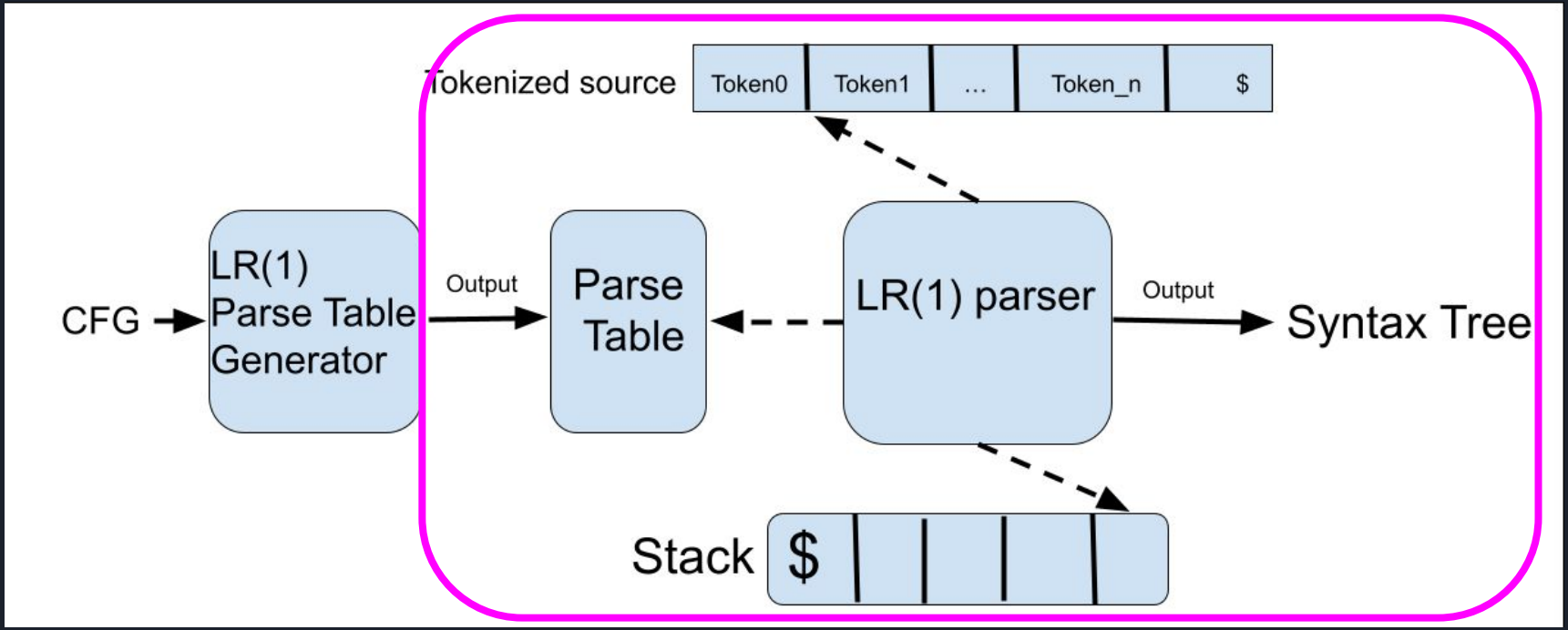
# LR(1) parser generator

- Stap 1



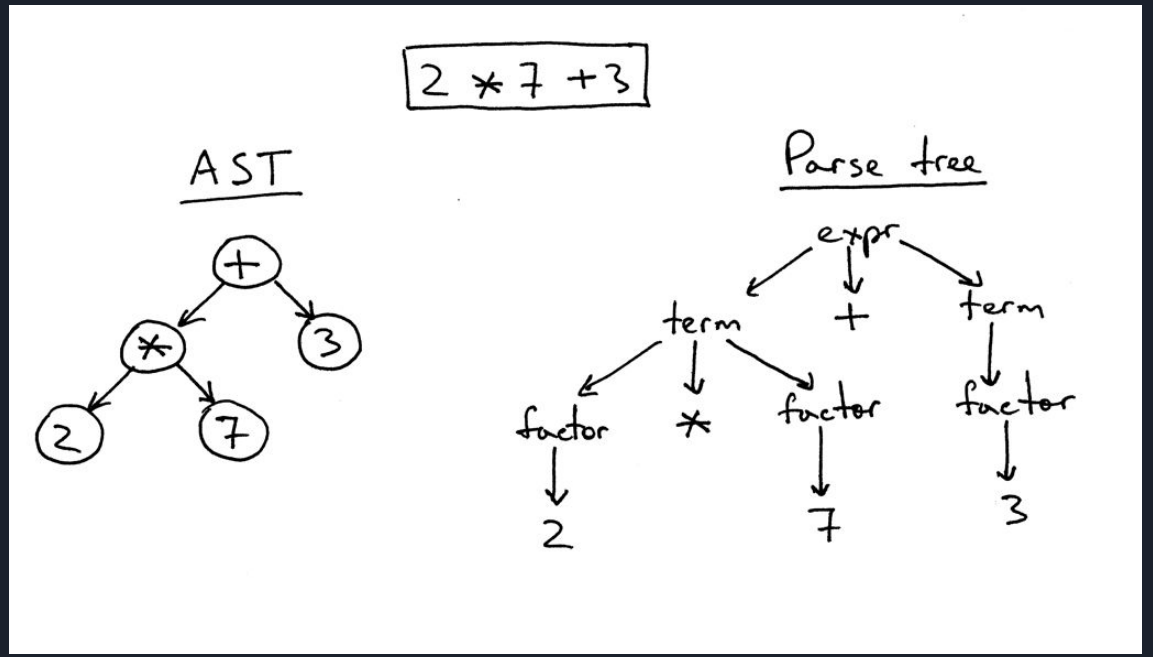
# LR(1) parser

- Stap 2



# Syntax Tree nabewerking:

- CST -> AST



# AST controleren op fouten:

# AST -> Javascript

Nodes to code

# Uitbreidingen

- Syntax tree view van de source code
- Bijkomend gesynchroniseerd met source view
- (Local) Variable rename mbv de syntax tree

# Einde