Wat doen taalingenieurs?

inleiding taalbouwkunde voor beginners

Tijs van der Storm, 1-10-2022

CWI
Centrum Wiskunde & Informatica

Inleiding

- Tijs van der Storm
- Groepsleider CWI SWAT:
  - ~~Special Weapons and Tactics~~ 😉
  - Software Analysis & Transformation
- Hoogleraar software engineering Groningen
- Onderzoek:
  - hoe kunnen we **betere programmeertalen** maken?
  - hoe kunnen we **beter** programmeertalen **maken**?

CWI
Centrum Wiskunde & Informatica

university of groningen

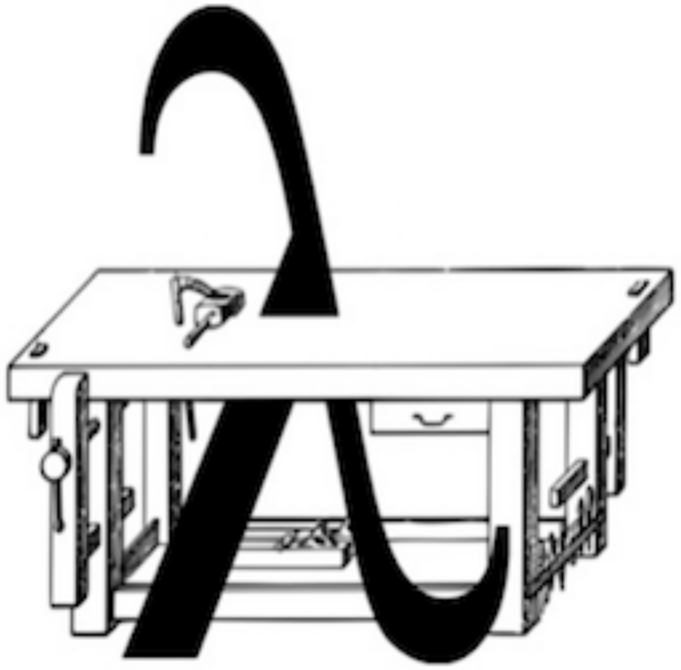
rascal-mpl.org

Wat doen taalingenieurs?

Language engineering

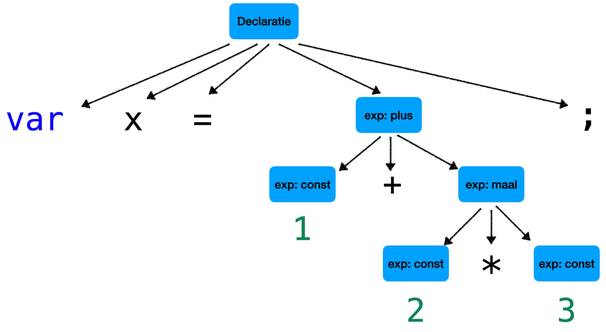
de “werktuigbouwkunde” van computertalen

- Hoe bouw je een computertaal?
- Op solide **principes** en **technieken** gestoeld
- We **weten** wat we doen, en **waarom**
- Bewust van **tradeoffs** en **omgevingsfactoren**
- We kunnen **uitspraken** doen over het resultaat
- Met de juiste **gereedschappen** op de juiste manier

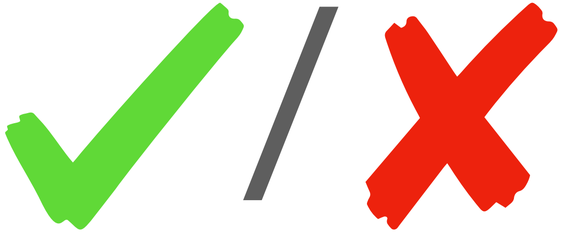


```
var x = 1 + 2 * 3;
```

Ontleden



Valideren



Uitvoeren

Resultaat

Ontleden

Piet koopt hoge schoenen

Ontleden

taalkundig / lexicaal / woordniveau

Piet koopt hoge schoenen

Ontleden

taalkundig / lexicaal / woordniveau

Piet koopt hoge schoenen

eigennaam

Ontleden

taalkundig / lexicaal / woordniveau

Piet koopt hoge schoenen

eigennaam werkwoord

Ontleden

taalkundig / lexicaal / woordniveau

Piet koopt hoge schoenen

eigennaam

werkwoord

bijvoeglijk naamwoord

Ontleden

taalkundig / lexicaal / woordniveau

Piet koopt hoge schoenen

eigennaam

werkwoord

bijvoeglijk naamwoord

zelfstandig naamwoord

Ontleden

redekundig / grammaticaal / zinsdeelniveau

Piet koopt hoge schoenen

Ontleden

redekundig / grammaticaal / zinsdeelniveau

Piet koopt hoge schoenen

onderwerp

Ontleden

redekundig / grammaticaal / zinsdeelniveau

Piet koopt hoge schoenen

onderwerp

persoons-
vorm

Ontleden

redekundig / grammaticaal / zinsdeelniveau

Piet koopt hoge schoenen

onderwerp

persoons-
vorm

lijdend voorwerp

Ontleden van computertaal

syntax analyse

```
var x = 1 + 2 * 3;
```

Ontleden van computertaal

lexicaal

```
var x = 1 + 2 * 3;
```

Ontleden van computertaal

lexicaal

```
var x = 1 + 2 * 3;
```

sleutelwoord
(keyword)

Ontleden van computertaal

lexicaal

```
var x = 1 + 2 * 3;
```

sleutelwoord (keyword)

naam (identifier)

Ontleden van computertaal

lexicaal

```
var x = 1 + 2 * 3;
```

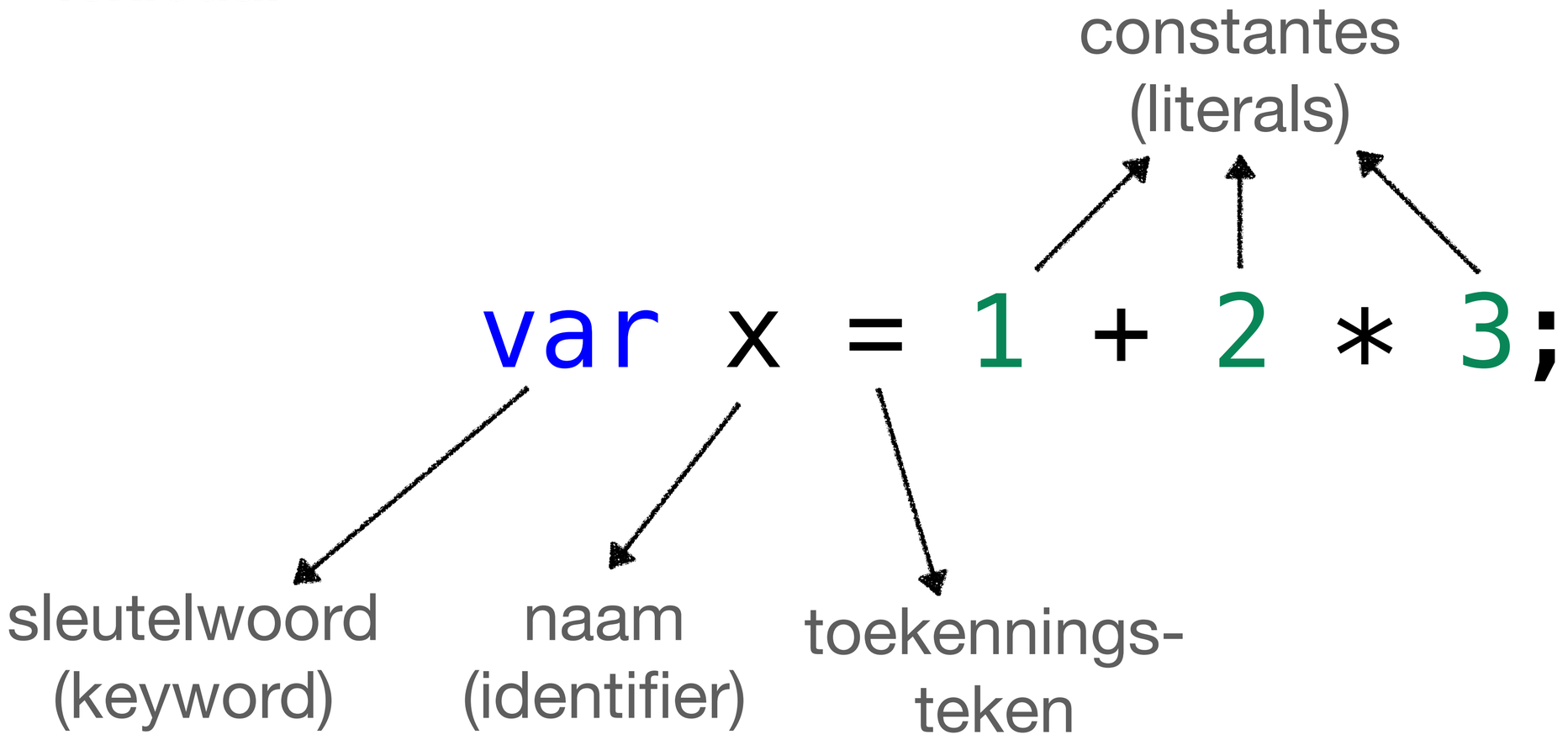
sleutelwoord (keyword)

naam (identifier)

toekennings-teken

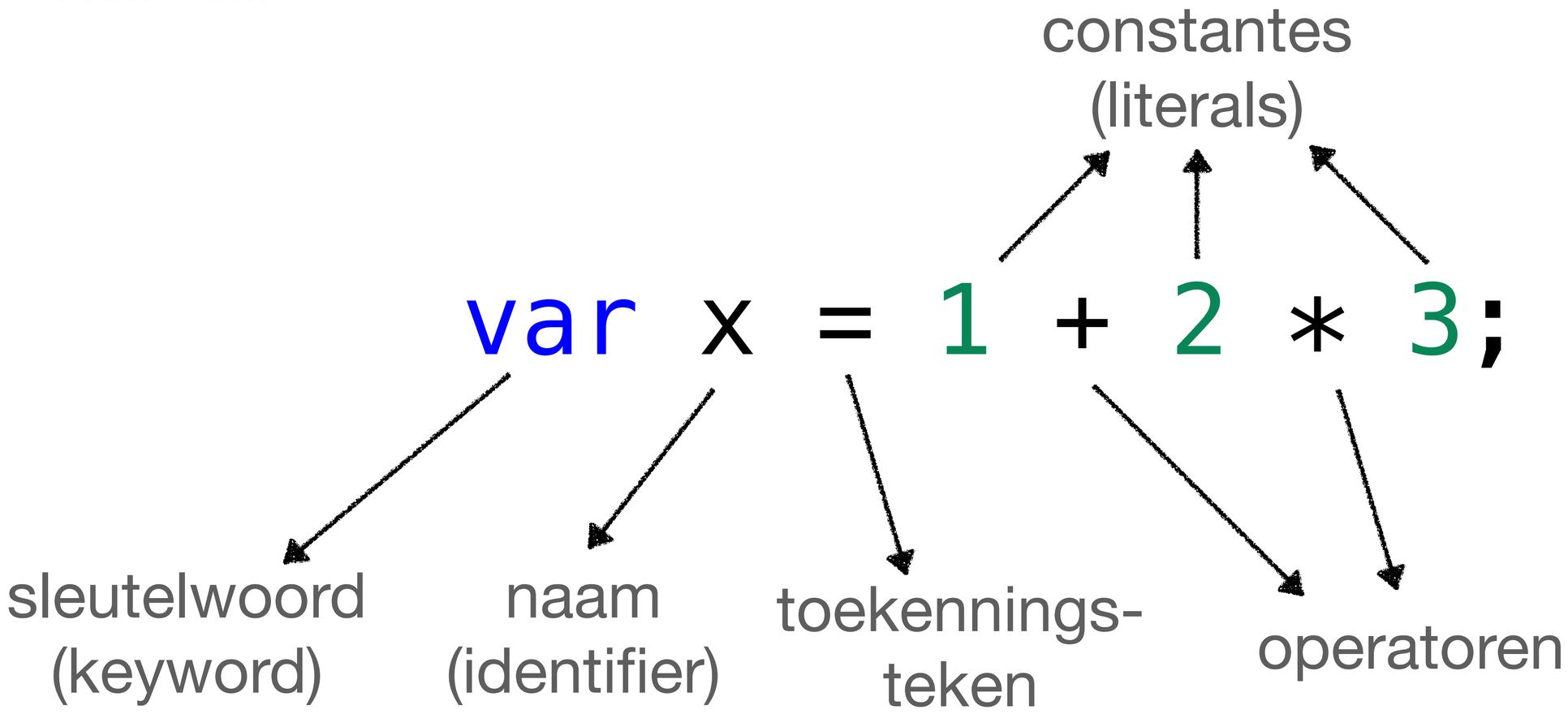
Ontleden van computertaal

lexicaal



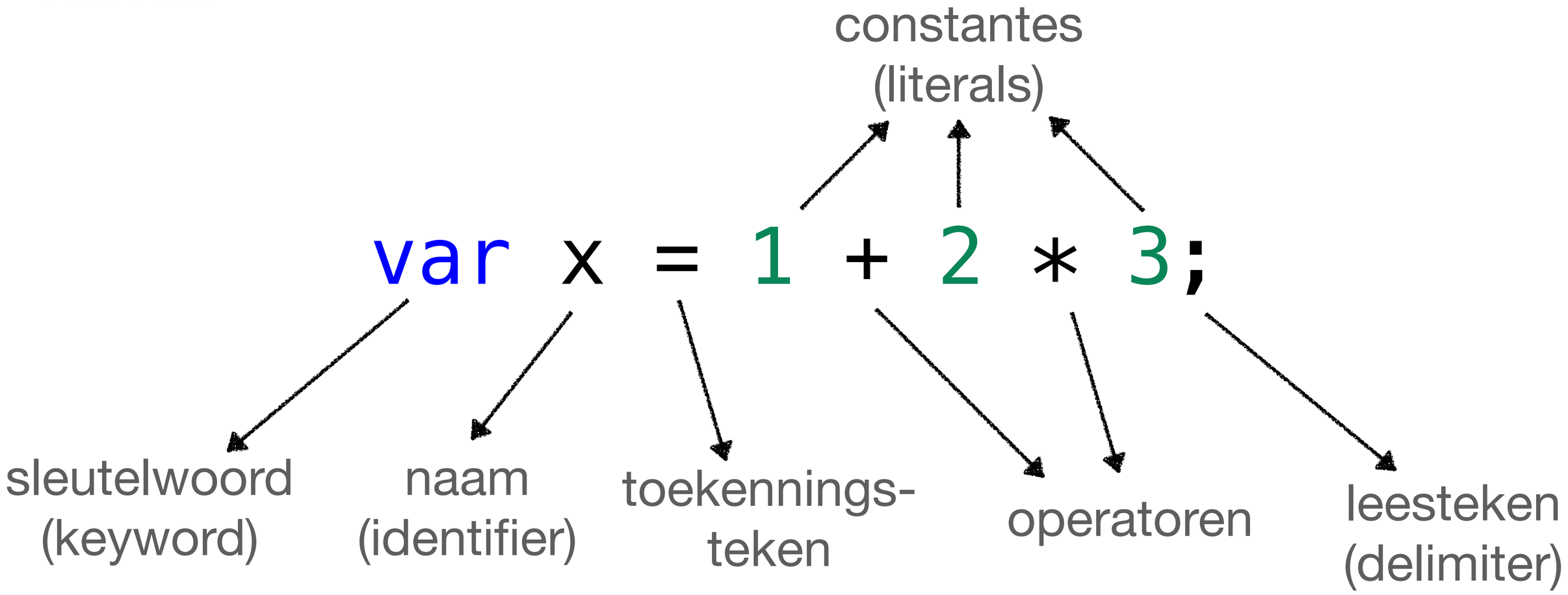
Ontleden van computertaal

lexicaal



Ontleden van computertaal

lexicaal



Ontleden van computertaal

grammaticaal

```
var x = 1 + 2 * 3;
```

NB: exp = “Expressie”

Ontleden van computertaal

grammaticaal

```
var x = 1 + 2 * 3;
```

exp exp exp

NB: exp = “Expressie”

Ontleden van computertaal

grammaticaal

```
var x = 1 + 2 * 3;
```

exp exp exp

exp

NB: exp = “Expressie”

Ontleden van computertaal

grammaticaal

```
var x = 1 + 2 * 3;
```

exp exp exp

exp

exp

NB: exp = “Expressie”

Ontleden van computertaal

grammaticaal

```
var x = 1 + 2 * 3;
```

exp exp exp

exp

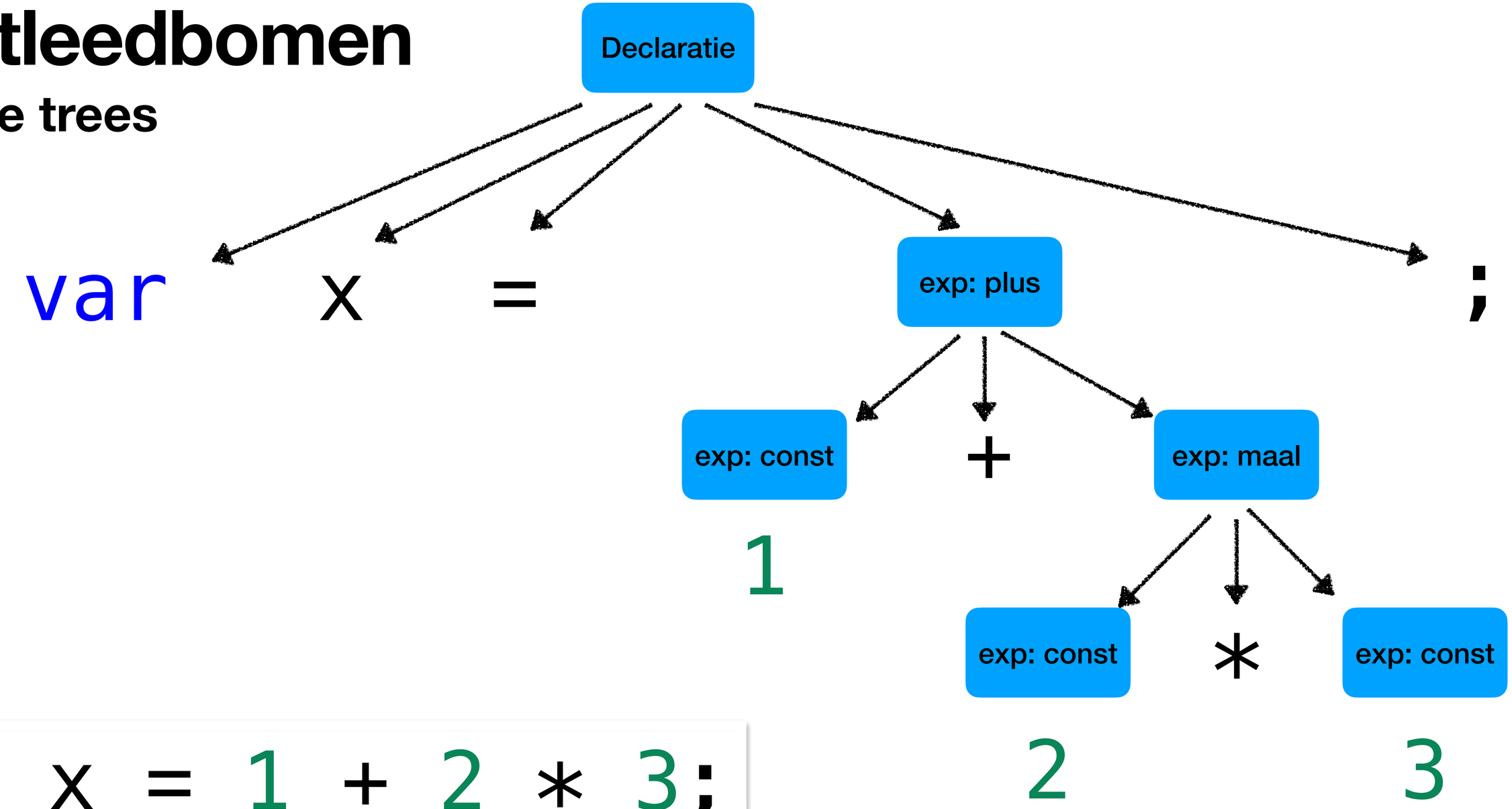
exp

declaratie

NB: exp = “Expressie”

Ontleedbomen

parse trees




```
var x = 1 + 2 * 3;
```

Language engineering onderzoek

syntax analyse

- Grammaticaformalismen
- Ontleedalgoritmes
  - LALR, LL(k), GLL, SGLR, Earley, PEG,…
- Foutdetectie, -afhandeling en -correctie
  - bv. ambiguïteiten
- Contextafhankelijke syntax
  - bv. indentatie

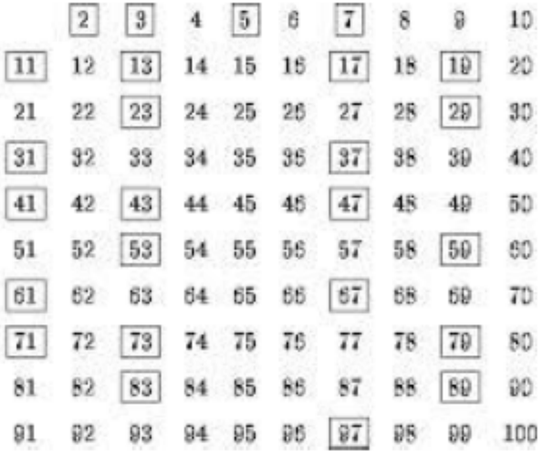
```
syntax Decl
  = "var" Name "=" Exp ";";

syntax Exp
  = Num
  | Name
  | left Exp "*" Exp
  > left Exp "+" Exp;

lexical Num = [0-9]+;

lexical Name = [a-z]+;
```

Zin en onzin: geldigheid

Caesar is een priemgetal

🤔

$\in$



	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Geldigheid

validiteit / “wellformedness”

```
var x = 1 + y * true;
```

Controleren op geldigheid

validiteit / “wellformedness” / naamanalyse / “type checking”

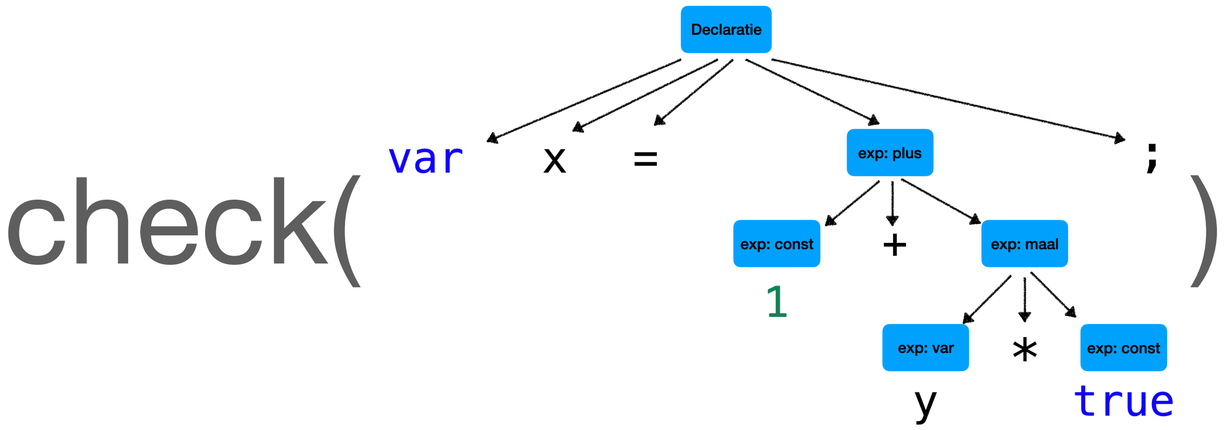
Variabele y is niet gedeclareerd

```
var x = 1 + y * true;
```

Vermenigvuldiging ongeldig
op waarheidswaarden

Controleren op geldigheid

validiteit / “wellformedness” / naamanalyse / “type checking”



- **Variabele y is niet gedeclareerd**
- **Vermenigvuldiging ongeldig op waarheidswaarden**

Language engineering onderzoek

- “Type checking”
  - ken types toe aan expressie
  - controleer of operaties geldig zijn
- Statische analyses
  - null pointers, dode code, “bugs”, geheugenlekken, veiligheid, etc.
- Formalismen voor verificatie
- “Model checking”

$$\frac{}{\Gamma, x : \tau \vdash x : \tau}\,Var$$

$$\frac{\Gamma, x : \tau_p \vdash t : \tau_r}{\Gamma \vdash (\lambda x : \tau_p . t) : \tau_p \to \tau_r}\,Lam$$

$$\frac{\Gamma \vdash t_f : \tau_p \to \tau_r \qquad \Gamma \vdash t_p : \tau_p}{\Gamma \vdash t_f\ t_p : \tau_r}\,App$$

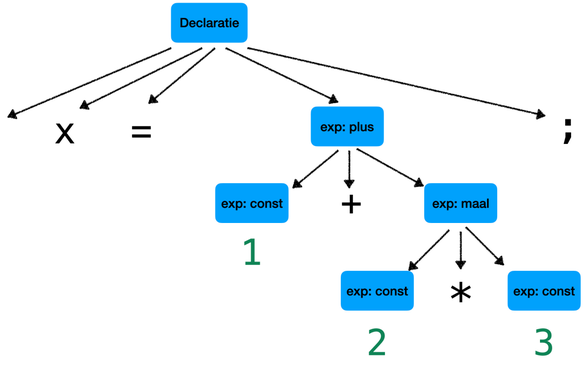
Maar wat betekent het?

semantiek / interpretatie / vertalen / executeren

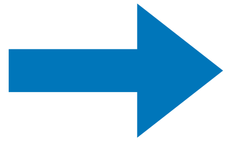
```
var x = 1 + 2 * 3;
```

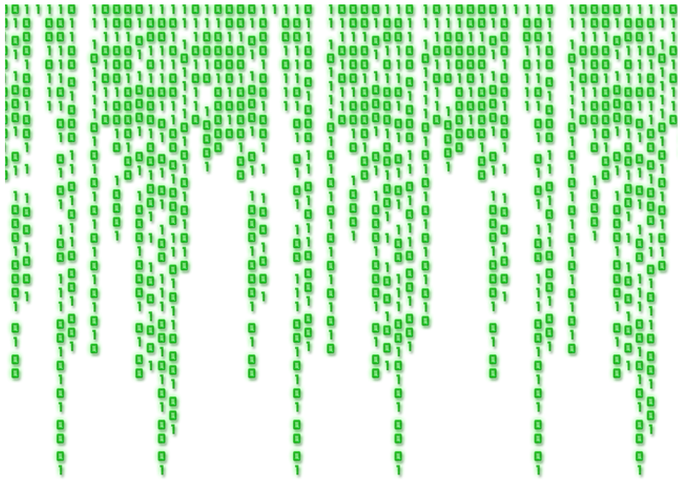
Vertalen en evalueren

compilers / interpreters

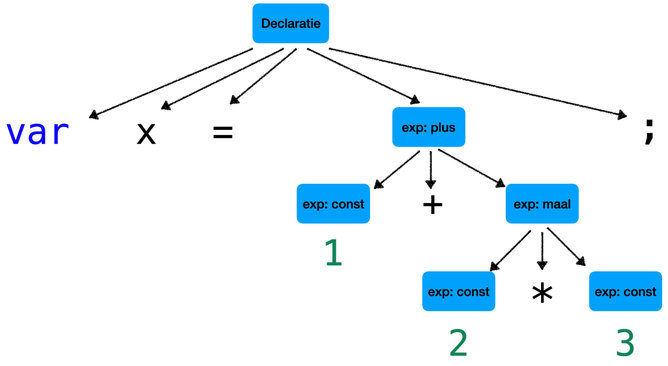
vertaal(

)

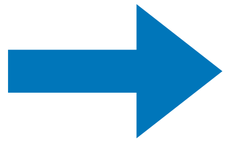




---

evalueer(

, *input*)



output

Evaluatie van expressies

definitie van een interpreter

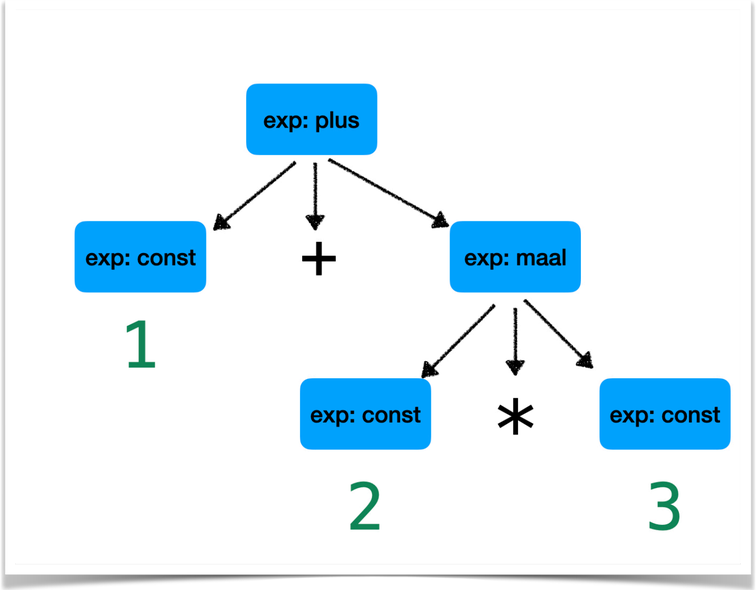
```
int eval(`n`, env) = toInt(n);

int eval(`x`, env) = lookup(env, x);

int eval(`l + r`, env)
   = eval(l, env) + eval(r, env);

int eval(`l * r`, env)
   = eval(l, env) * eval(r, env);
```

```
syntax Exp
  = Num
  | Name
  | left Exp "*" Exp
  > left Exp "+" Exp;

lexical Num = [0-9]+;

lexical Name = [a-z]+;
```

De expressiegrammatica

Evaluatie van expressies

evaluatie trace




```
eval(`1 + 2 * 3`, env)

   = eval(`1`, env) + eval(`2 * 3`, env)

       = 1            +
                          eval(`2`, env) * eval(`3`, env)

                            = 2        *       = 3

                                       = 6

                     = 7
```

Vertalen van expressies

naar instructies van een stack machine

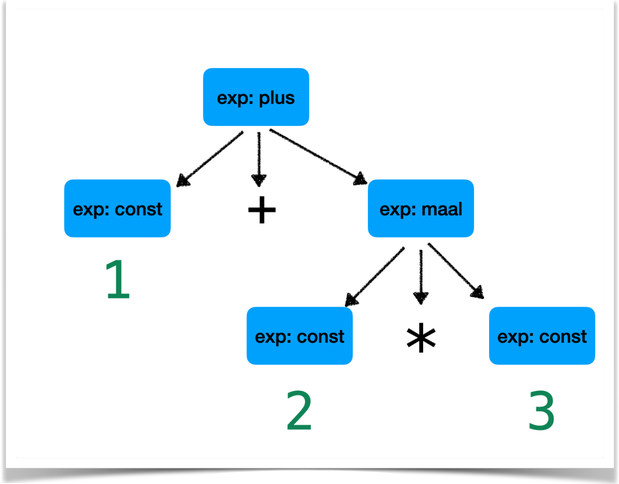
De doeltaal (target language)

```
syntax Calc = Instr*;

syntax Instr
  = "push" Num
  | "add"
  | "mul";
```

De vertaler (compiler)

```
Calc trans(`n`) = `push n`;

Calc trans(`l + r`, env)
   = trans(`l`) ++ trans(`r`) ++ `add`;

Calc trans(`l * r`, env)
  = trans(`l`) ++ trans(`r`) ++ `mul`;
```

Vertalen van expressies




```
trans(`1 + 2 * 3`)

   = trans(`1`, env) ++ trans(`2 * 3`)                    ++ `add`

       = `push 1`    ++
                        trans(`2`) ++ trans(`3`)   ++ `mul`

                         = `push 2`     = `push 3`

                                 = `push 2 push 3 mul`

                    = `push 1 push 2 push 3 mul add`
```

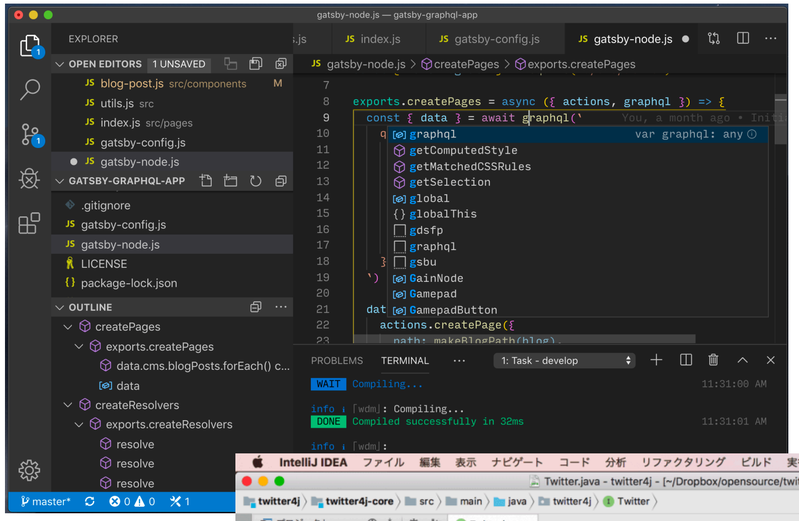
Language engineering onderzoek

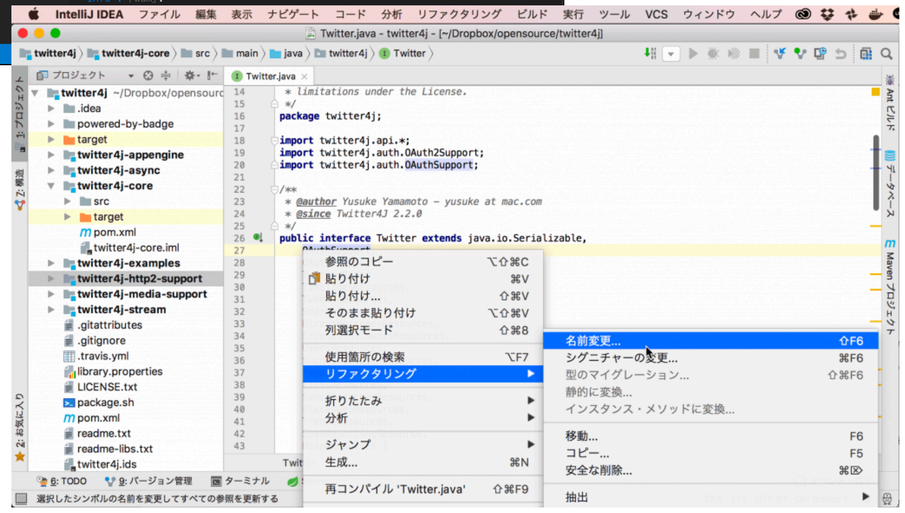
- Transformatietalen
- Compileroptimalisaties
- Semantiekformalismen om eigenschappen te bewijzen.
- Virtual machines
- Interpreter frameworks / patterns

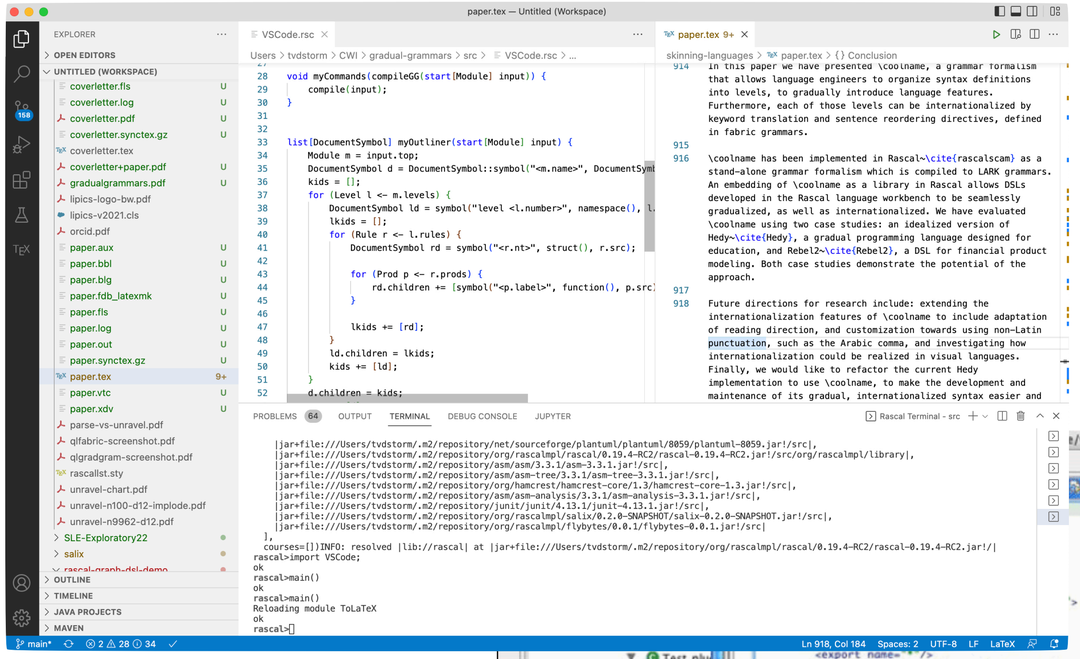
```
Expr: (Expr) `1 - 2 * 3 - 3`
rascal>redexSteps(e);
 ├─ 1 - 2 * 3 - 3  —mul→ 1 - 6 - 3
 │   ├─ 1 - 6 - 3  —sub→ -5 - 3
 │   │   └─ -5 - 3  —sub→ -8
 │   └─ 1 - 6 - 3  —sub→ 1 - 3
 │       └─ 1 - 3  —sub→ -2
 ├─ 1 - 2 * 3 - 3  —sub→ -1 * 3 - 3
 │   ├─ -1 * 3 - 3  —sub→ -1 * 0
 │   │   └─ -1 * 0  —mul→ 0
 │   └─ -1 * 3 - 3  —mul→ -3 - 3
 │       └─ -3 - 3  —sub→ -6
 └─ 1 - 2 * 3 - 3  —sub→ 1 - 2 * 0
     ├─ 1 - 2 * 0  —sub→ -1 * 0
     │   └─ -1 * 0  —mul→ 0
     └─ 1 - 2 * 0  —mul→ 1 - 0
         └─ 1 - 0  —sub→ 1
ok
```

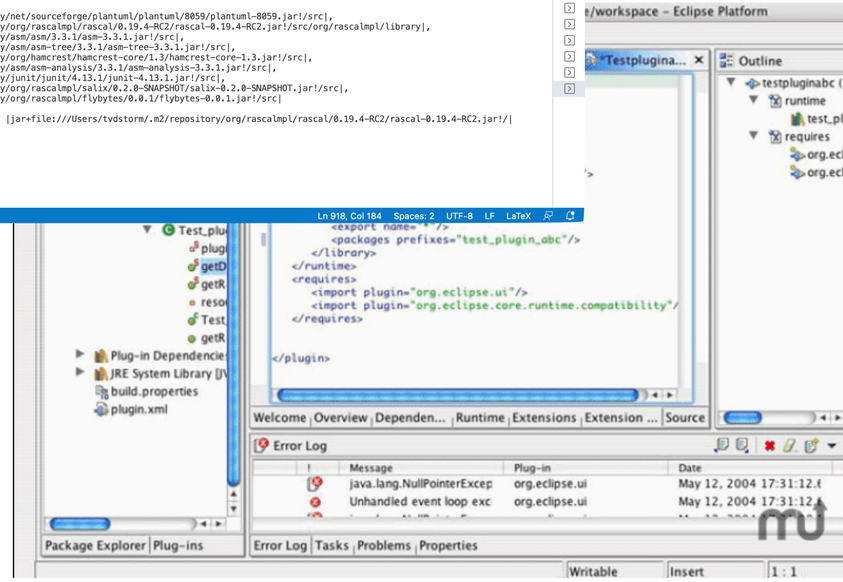
Geïntegreerde ontwikkelomgevingen

“Integrated Development Environments (IDEs)”



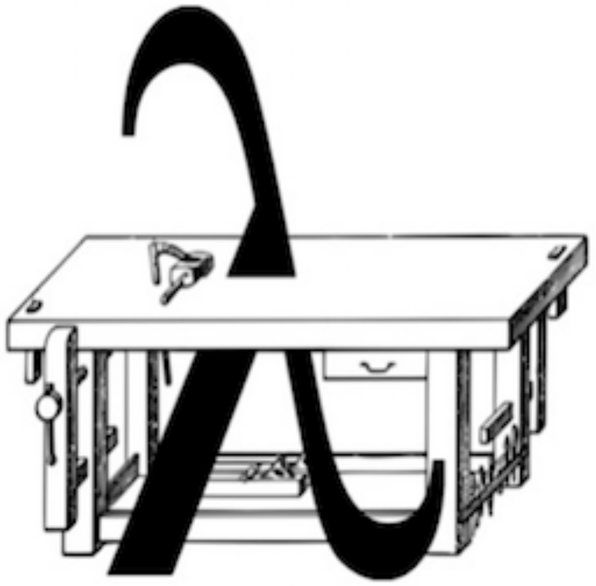






Language Workbenches

werkbanken voor taalontwerp en implementatie



- Geïntegreerde **meta** omgevingen
- Power tools voor language engineering
  - van syntax analyse tot IDE ondersteuning
- CWI SWAT: **metaprogrammeertaal** Rascal
- Ontworpen voor alles wat met broncode te maken heeft, inclusief talen bouwen!

rascal-mpl.org

Zijn er niet al genoeg programmeertalen?

of: het belang van meer talen

- Mainstream talen zijn *general purpose*: **grote talen** die allemaal ongeveer **even goed** zijn in **bijna alles**.
- Domain-specific languages (DSL): **kleine talen** die **kampioen** zijn in specifieke **probleemdomeinen**.
  - “economy of expression”, dicht bij probleemdomein.
  - -> meer mogelijk maken met minder code.
- Language engineering draagt bij aan het kosteneffectief bouwen van DSLs
- SWAT.engineering: language engineering spin-off

https://www.tiobe.com/tiobe-index/

Sep 2022	Sep 2021	Change	Programming Language
1	2	^	Python
2	1	v	C
3	3		Java
4	4		C++
5	5		C#
6	6		Visual B
7	7		JavaScr
[illegible]	8		Assemb languag
[illegible]	10	^	SQL
10	9	v	PHP

Wat doen taalingenieurs?

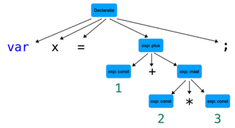
CWI
Centrum Wiskunde & Informatica

Language engineering
de “werktuigbouwkunde” van computertalen

- Hoe bouw je een computertaal?
- Op solide **principes** en **technieken** gestoeld
- We **weten** wat we doen, en **waarom**
- Bewust van **tradeoffs** en **omgevingsfactoren**
- We kunnen **uitspraken** doen over het resultaat
- Met de juiste **gereedschappen** op de juiste manier

```
var x = 1 + 2 * 3;
```

Ontleden



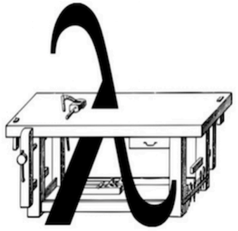
Valideren ✓ / ✗

Uitvoeren Resultaat

rascal-mpl.org

Language Workbenches
werkbanken voor taalontwerp en implementatie

- Geïntegreerde **meta** omgevingen
- Power tools voor language engineering
  - van syntax analyse tot IDE ondersteuning
- CWI SWAT: **metaprogrammeertaal** Rascal
- Ontworpen voor alles wat met broncode te maken heeft, inclusief talen bouwen!



rascal-mpl.org

General purpose vs. domain-specific

www.swat.engineering