intelligent views

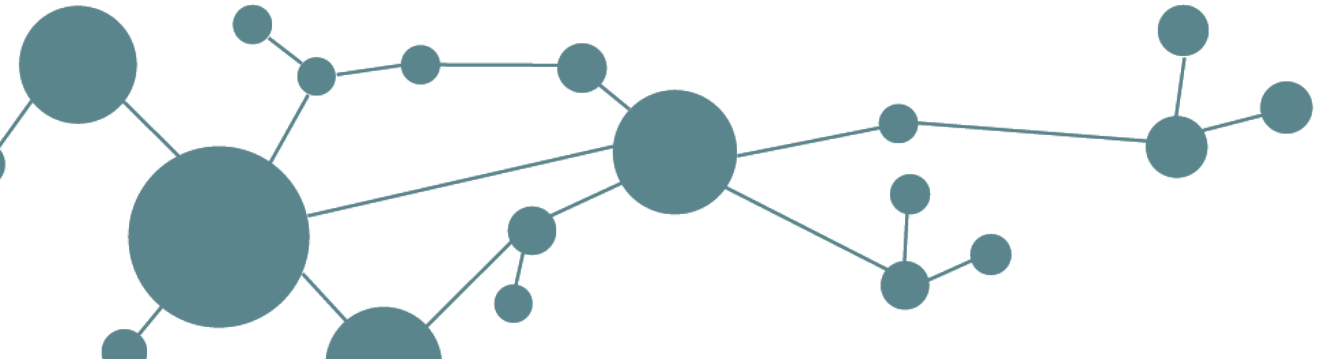
K-Infinity

JavaScript for VW Applications

Holger Kleinsorgen & Jan Schümmer

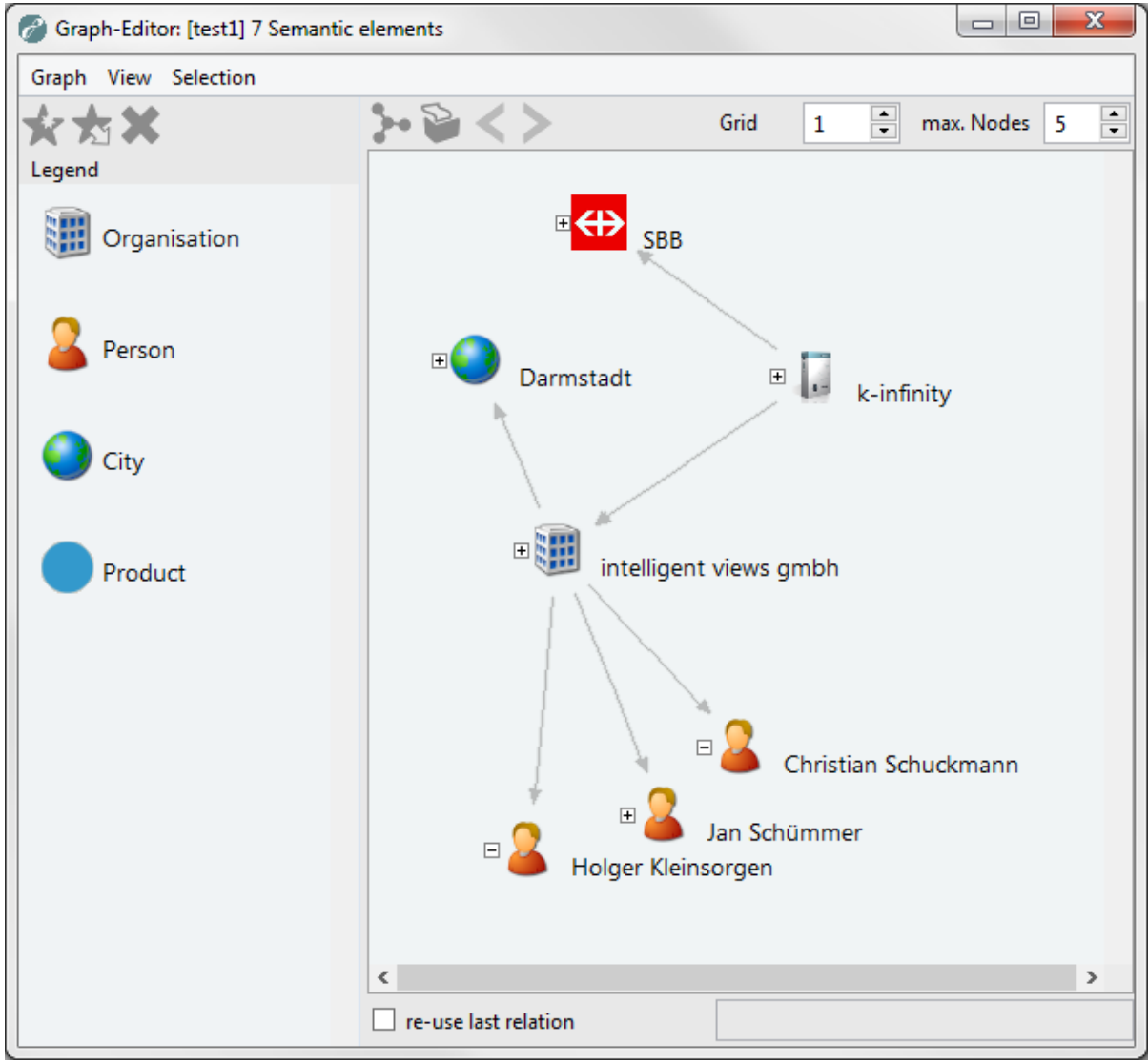
intelligent views, Darmstadt, Germany

ESUG 2014 – Cambridge, UK



K-Infinity - our core product

- semantic database
- modelling
- querying
- data integration
- web frontend (Java)
- REST services
- highly configurable



→ need for scripting throughout the system

	First try: KScript	Smalltalk sandbox	JavaScript
easy to communicate	☹	☺	☺☺
easy to learn	☹	☺	☺
universal / wide spread	☹☹	😐	☺☺
suitable	😐	☺	☺
flexible	☹	☺	☺

Anything reusable out there?

only one dead ST implementation

3rd party C libraries

- tied to web environments
- uncertain future

Mozilla Rhino

- Java implementation would need JNI
- lack of flexibility for host object integration

We wanted real Smalltalk Objects without additional runtimes

let's implement our own parser and runtime

What is important?

- conform to the specs
- easy to integrate host objects
- maintainable

What is not that important

- performance

→ simply follow the 250 pages of the ECMA 5 spec

Parser

- PetitParser
  - PEG + Memoizing + Stuff, Smalltalk notation
  - Tweaked for more precise error locations
- separated into lexical and syntactical part
  - follows structure of ECMA spec
  - adapted for PEG (no left recursions, alternatives parse first match)
- quirks due to spec
  - strict mode
  - distinguish regex literals from expressions
  - automatic semicolon insertion

Parser

VariableStatement :
    var *VariableDeclarationList* ;

```
variableStatement

    ^ 'var' jseTokenParser , variableDeclarationList , automaticSemicolon
        jseAction: [: elements : sourceInterval |
            self factory createVariableStatement: ( elements at: 2 ) sourceInterval: sourceInterval
        ]
```

Runtime

- Everything represented by dedicated objects
  - JSEAbstractObject (Javascript objects and primitives)
  - JSEExecutionEnvironment + JSEExecutionContext
  - JSEBinding, JSEReference, JSEProperty
  - Implement algorithms of the spec
  - VW VM garbage collection
- Language core implemented by JSERuntimeImplementation
- Syntax tree transformed to code blocks
- Code blocks are evaluated in an execution context
  - Interpreted and thus quite slow

ECMAScript Language Specification 5.1, page 76

11.7.1 The Left Shift Operator (<<)

Performs a bitwise left shift operation on the left operand by the amount specified by the right operand.

The production *ShiftExpression* : *ShiftExpression* **<<** *AdditiveExpression* is evaluated as follows:

1. Let *lref* be the result of evaluating *ShiftExpression*.
2. Let *lval* be GetValue(*lref*).
3. Let *rref* be the result of evaluating *AdditiveExpression*.
4. Let *rval* be GetValue(*rref*).
5. Let *lnum* be ToInt32(*lval*).
6. Let *rnum* be ToUint32(*rval*).
7. Let *shiftCount* be the result of ma
   & 0x1F.
8. Return the result of left shifting *ln*

11.7.2 The Signed Right Shift Oper

Performs a sign-filling bitwise right s
operand

```
JSEAbstractObject>>leftShift: anotherObject
    " 11.7.1 The Left Shift Operator ( << ) "

    | leftNumber rightNumber shiftCount |
    leftNumber := self toSignedInteger32Value.
    rightNumber := anotherObject toUnsignedInteger32Value.
    shiftCount := rightNumber bitAnd: 16r1f.
    ^ (leftNumber bitShift: shiftCount) jseToSignedInteger32Value
```

ECMA conformance test suite

- about 11500 unit tests
- current result: 313 failures, 95 errors (4 %)
- browsers: between 8 and 52 failures / errors (0.06 % – 0.45 %)
- test cases helped to find the nifty / exotic parts
- decided to accept some failures for the time being

Demo time

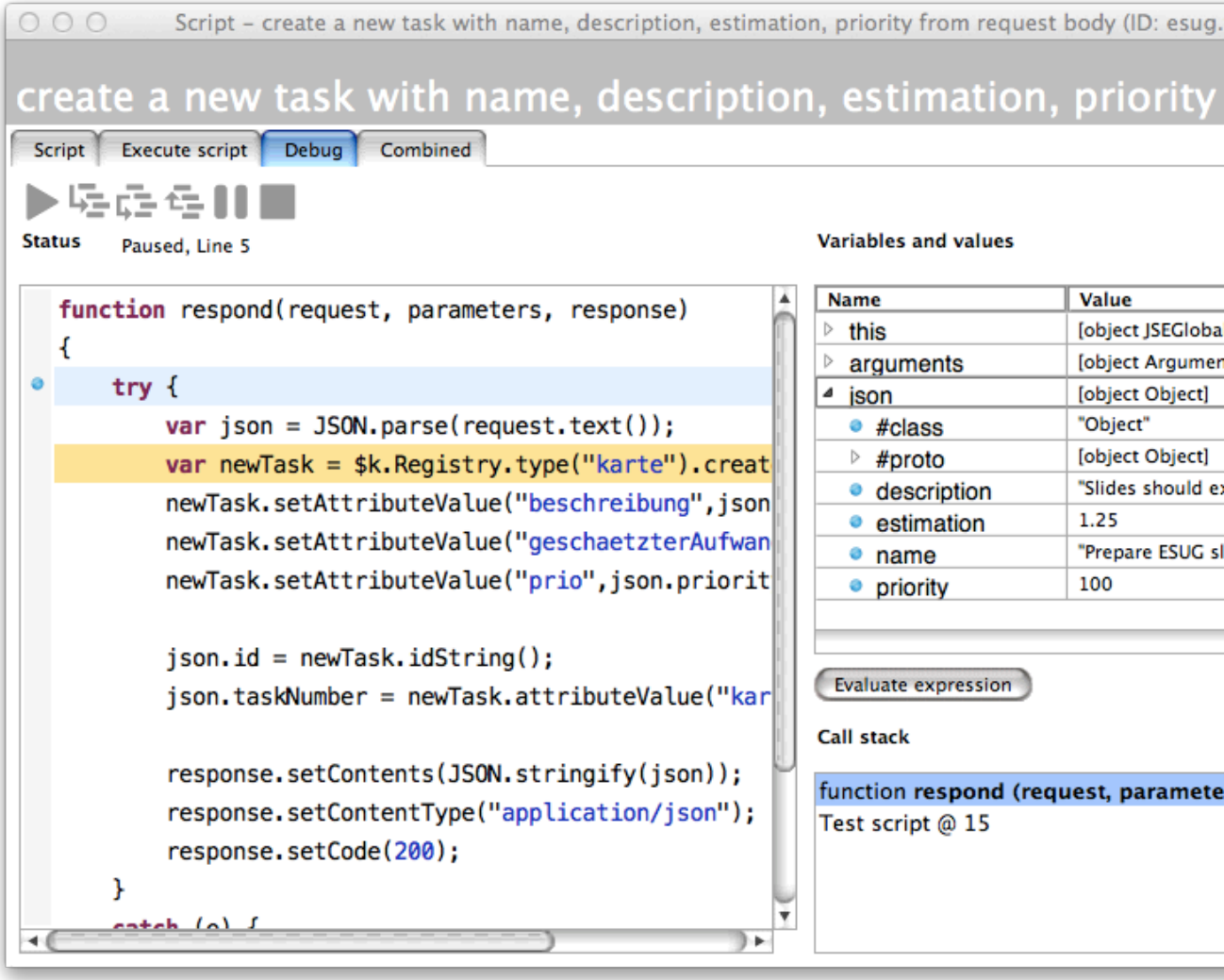
K-Infinity Integration

Usage Example

- definition of a REST service

Semantic Net API

Debugger



Ups and Downs

The good parts

- (mostly) works as expected
- implementation was straight forward
- use cool JS libraries
- fast enough for our use cases

The cheesy parts

- parser hacks
- the "usual" browser objects are not available
- took some time

Wish list

- switch to single grammar
- switch to VW PEG Parser
  - nicer to read due to DSL, easier to maintain
  - already includes JS grammar, needs some tweaks though
- byte code compiler
  - only as a pet project, not required currently
- all test cases pass
- ECMA 6 (especially collections)

JavascriptEngine-Bundle

MIT License

Public Repository

{h.kleinsorgen|j.schuemmer}@i-views.de

Thank you for your attention