

Implementering av en Lua parser

Oskar Schöldström

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	1234
Författare:	Oskar Schöldström
Arbetets namn:	Implementering av en Lua parser
Handledare (Arcada):	...
Uppdragsgivare:	
Sammandrag:	
a a	
Nyckelord:	parser, javascript, lua, lexer, tolk
Sidantal:	13
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information and Media Technology
Identification number:	1234
Author:	Oskar Schöldström
Title:	Implementing a Lua parser
Supervisor (Arcada):	...
Commissioned by:	
Abstract:	
a a	
Keywords:	parser, javascript, lua, lexer, lexical analysis, tokenizer, scanner, lexical analyzer, syntactic analysis
Number of pages:	13
Language:	Swedish
Date of acceptance:	

OPINNÄYTE	
Arcada	
Koulutusohjelma:	Informaatio- ja mediatekniikka
Tunnistenumero:	1234
Tekijä:	Oskar Schöldström
Työn nimi:	Implementing a Lua parser
Työn ohjaaja (Arcada):	...
Toimeksiantaja:	
Tiivistelmä: Lorem ipsum dolor sit amet, consectetur adipiscing elit. Maecenas molestie, nisi sed consectetur tincidunt, ante libero euismod nulla, eget commodo ligula purus eu nibh. Ut enim risus, congue sed viverra ut, tempor vel velit. Quisque consectetur tincidunt scelerisque. Donec nec mollis leo. Donec laoreet purus a massa ultrices egestas. Ut quis neque nisi. In hac habitasse platea dictumst. Suspendisse suscipit, ante eget venenatis molestie, elit nulla aliquam ligula, sit amet consectetur tortor magna eu ipsum. Pellentesque tempor tincidunt arcu, eget ultrices lorem commodo a.	
Avainsanat:	parser, javascript, lua, lexer, lexical analysis, tokenizer, scanner, lexical analyzer, syntactic analysis
Sivumäärä:	13
Kieli:	Ruotsi
Hyväksymispäivämäärä:	

INNEHÅLL

Förord	8
1 Inledning	10
1.1 Syfte och målsättning	10
1.2 Avgränsning	10
2 Teori	10
2.1 Lexer	10
2.1.1 Tokenization	10
2.2 Syntaktisk analys	10
2.2.1 Kontextfri grammatik	10
2.2.2 Backaus-Naur Form	10
2.2.3 LR-parser	10
2.2.4 LL-parser	10
2.2.5 AST	10
2.3 Parser generators	10
3 Design	10
3.1 Krav	10
3.1.1 Prestanda	10
3.1.2 Utvecklingsmiljö	10
3.1.3 Framtida utveckling	10
3.2 Design	10
3.2.1 Flöde	10
3.2.2 JavaScript	10
4 Implementation	10
4.1 Grammatik	10
4.1.1 Literals	10
4.1.2 Expressions	10
4.1.3 Assignments	10
4.2 Output	10
4.2.1 AST	10
4.3 Optimering	10
4.3.1 Mikro optimering	10
4.3.2 Datastrukturer	10
4.3.3 Trie	10
5 Diskussion	10
Källor	11
Bilaga - Källkod	12

TABELLER

FIGURER

FÖRKORTNINGAR OCH DEFINITIONER

LL parser asdasdasda

LR parser asdasdasda

Syntax asdasdasda

BNF asdasdasda

Semantik asdasdasda

FÖRORD

....

1 INLEDNING

1.1 Syfte och målsättning

1.2 Avgränsning

2 TEORI

2.1 Lexer

2.1.1 Tokenization

2.2 Syntaktisk analys

2.2.1 Kontextfri grammatik

2.2.2 Backaus-Naur Form

2.2.3 LR-parser

2.2.4 LL-parser

2.2.5 AST

2.3 Parser generators

3 DESIGN

3.1 Krav

3.1.1 Prestanda

10

3.1.2 Utvecklingsmiljö

KÄLLOR

BILAGA - LUA GRAMMATIK

$\langle chunk \rangle$	$::= \langle block \rangle$
$\langle block \rangle$	$::= \{ \langle stat \rangle \} [\langle retstat \rangle]$
$\langle stat \rangle$	$::= ;$ $\langle varlist \rangle = \langle explist \rangle$ $\langle functioncall \rangle$ $\langle label \rangle$ break goto Name do $\langle block \rangle$ end while $\langle exp \rangle$ do $\langle block \rangle$ end repeat $\langle block \rangle$ until $\langle exp \rangle$ if $\langle exp \rangle$ then $\langle block \rangle$ { elseif $\langle exp \rangle$ then $\langle block \rangle$ } [else $\langle block \rangle$] end for Name = $\langle exp \rangle$, $\langle exp \rangle$ [, $\langle exp \rangle$] do $\langle block \rangle$ end for $\langle namelist \rangle$ in $\langle explist \rangle$ do $\langle block \rangle$ end function funcname $\langle funcbody \rangle$ local $\langle function \rangle$ Name $\langle funcbody \rangle$ local $\langle namelist \rangle$ [= $\langle explist \rangle$]
$\langle retstat \rangle$	$::= \textbf{return} [\langle explist \rangle] [;]$
$\langle label \rangle$	$::= :: \text{Name} ::$
$\langle funcname \rangle$	$::= \text{Name} \{ . \text{Name} \} [: \text{Name}]$
$\langle varlist \rangle$	$::= \langle var \rangle \{ , \langle var \rangle \}$
$\langle var \rangle$	$::= \text{Name} \mid \langle prefixexp \rangle [\langle exp \rangle] \mid \langle prefixexp \rangle . \text{Name}$
$\langle namelist \rangle$	$::= \text{Name} \{ , \text{Name} \}$
$\langle explist \rangle$	$::= \langle exp \rangle \{ , \langle exp \rangle \}$
$\langle exp \rangle$	$::= \textbf{nil} \mid \textbf{false} \mid \textbf{true} \mid \text{Number} \mid \text{String} \mid \dots \mid \langle functiondef \rangle \mid \langle prefixexp \rangle$ $\langle tableconstructor \rangle \mid \langle exp \rangle \langle binop \rangle \langle exp \rangle \mid \langle unop \rangle \langle exp \rangle$
$\langle prefixexp \rangle$	$::= \langle var \rangle \mid \langle functioncall \rangle \mid (\langle exp \rangle)$
$\langle functioncall \rangle$	$::= \langle prefixexp \rangle \langle args \rangle \mid \langle prefixexp \rangle : \text{Name} \langle args \rangle$
$\langle args \rangle$	$::= ([\langle explist \rangle]) \mid \langle tableconstructor \rangle \mid \text{String}$
$\langle functiondef \rangle$	$::= \textbf{function} \langle funcbody \rangle$
$\langle funcbody \rangle$	$::= ([\langle parlist \rangle]) \langle block \rangle \textbf{end}$
$\langle parlist \rangle$	$::= \langle namelist \rangle [, \dots] \mid \dots$
$\langle tableconstructor \rangle$	$::= \{ [\langle fieldlist \rangle] \}$
$\langle fieldlist \rangle$	$::= \langle field \rangle \{ \langle fieldsep \rangle \langle field \rangle \} [\langle fieldsep \rangle]$
$\langle field \rangle$	$::= [\langle exp \rangle] = \langle exp \rangle \mid \text{Name} = \langle exp \rangle \mid \langle exp \rangle$
$\langle fieldsep \rangle$	$::= , \mid ;$
$\langle binop \rangle$	$::= + \mid - \mid * \mid / \mid ^ \mid \% \mid .. \mid < \mid <= \mid > \mid >= \mid == \mid ~= \mid \textbf{and} \mid \textbf{or}$
$\langle unop \rangle$	$::= - \mid \textbf{not} \mid \#$

BILAGA - KÄLLKOD