

# The WyScript Language Specification

David J. Pearce  
School of Engineering and Computer Science  
Victoria University of Wellington, New Zealand  
[djp@ecs.vuw.ac.nz](mailto:djp@ecs.vuw.ac.nz)

December 4, 2013

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
1.1	Overview . . . . .	2
1.2	Goals . . . . .	2
1.3	History . . . . .	2
<b>2</b>	<b>Syntax</b>	<b>3</b>

# **Chapter 1**

## **Introduction**

### **1.1 Overview**

### **1.2 Goals**

### **1.3 History**

$prgrm ::= (d)^*$	// WyScript Program
$d ::=$	// Declarations
type ID is t newline	// Type Declarations
constant ID is e newline	// Constant Declarations
t ID ( [ t ID ( , t ID ) <sup>*</sup> ] ) : B newline	// Function Declarations
$B ::= ( \text{indent } m )^*$	// Blocks
$m ::=$	// Statements
return [ e ] newline	// Return Statements
print e newline	// Print Statements
if e : newline B [ else : newline B ]	// If Statements
while e : newline B	// While Statements
for ID in e : newline B	// For Statements
ID ( [ e ( , e ) <sup>*</sup> ] ) newline	// Invoke Statements
t n [ = e ] newline	// Variable Declaration Statements
( n   e <sub>1</sub> [ e <sub>2</sub> ]   e . ID ) = e newline	// Assignment Statements

Figure 2.1: Syntax for Declarations, Blocks and Statements

## Chapter 2

# Syntax

$e ::=$	<i>// Expressions</i>
$b$	<i>// Boolean constants</i>
$i$	<i>// Integer constants</i>
$r$	<i>// Rational constants</i>
$c$	<i>// Character constants</i>
$s$	<i>// String constants</i>
$n$	<i>// Variables</i>
$\boxed{\text{null}}$	<i>// Null constant</i>
$\boxed{( e )}$	<i>// Bracketed expressions</i>
$\boxed{  e  }$	<i>// Size expressions</i>
$e_1 \boxed{[ e_2 ]}$	<i>// Index Expressions</i>
$e \boxed{[ e_1 .. e_2 ]}$	<i>// Range Expressions</i>
$e \boxed{. ID}$	<i>// Access Expressions</i>
$\boxed{ID ( [ \bar{e} ( , e )^* ] )}$	<i>// Invocation expressions</i>
$\boxed{[ [ e_1 ( , e_i )^* ] ]}$	<i>// List expressions</i>
$\boxed{\{ [ ID : e_1 ( , ID_i : e_i )^* ] \}}$	<i>// Record Assignment expressions</i>

Figure 2.2: Syntax for Expressions

	$n$	<code>is</code>	$t$	// Instance of Expressions						
	<code>(</code>	$t$	<code>)</code>	$n$	// Cast Expressions					
	<code>(</code>	<code>-</code>	<code>!</code>	<code>&amp;</code>	<code>*</code>	<code>)</code>	$e$	// Unary expressions		
	$e_1$	<code>[</code>	$op^2$	$e_2$	<code>]</code>	$^+$	// Binary expressions			
$op^2$	<code>::=</code>	<code>+</code>	<code>-</code>	<code>*</code>	<code>/</code>	<code>%</code>	<code>++</code>	// Binary Operators		
	<code>&amp;&amp;</code>	<code>  </code>	<code>&gt;</code>	<code>&gt;=</code>	<code>&lt;</code>	<code>&lt;=</code>	<code>==</code>	<code>!=</code>	// Binary Comparisons	
$t$	<code>::=</code>								// Types	
	<code>ID</code>								// User-Defined Types	
	$t$	<code>[</code>	<code> </code>	$t$	<code>]</code>	$^+$			// Union Types	
	<code>[</code>	$t$	<code>]</code>							// List Types
	<code>{</code>	$t_1$	$n_1$	<code>(</code>	$t_i$	$n_i$	<code>*</code>	<code>}</code>	// Record Types	
	<code>void</code>	<code>null</code>	<code>bool</code>	<code>char</code>	<code>string</code>	<code>int</code>	<code>real</code>	// Base Types		

Figure 2.3: Syntax for Operations and Types