Language Summary

Basic Features of Boo

Structure of a Boo Script	`"""module docstring"""`
	
	`namespace My.NameSpace`
	
	`import Assembly.Reference`
	
	`class MyClass:`
	`    pass`
	
	`def domyfunction(it):`
	`    print(it)`
	
	`domyfunction(3)`
	
	`[assembly: AssemblyTitle('foo')]`
	
	`// [Module] attribute defines "global" class`
	`// Entry point: def Main(argv as (string)):`

Import	`import System`
	`import System.Console // Import type`
	`import Some.Namespace from Weird.Assembly.Name`
	`import Gtk from "gtk-sharp"`

Comments	`# a comment`
	`// a comment`
	`/* a comment */`
	`/* /* a comment */ */`

Builtin Functions Summary	`array(enumerable)`
	`array(type, collection)`
	`array(type, enumerable)`
	`array(type, size)`
	`enumerate(enumerable) // Returns index and the object itself`
	`gets`
	`gets()`
	`iterator(enumerable) as IEnumerable // Usually not necessary`
	`join(enumerable)`
	`join(enumerable, seperator as Char)`
	`map(enumerable, function)`
	`matrix(elementType, length as (int))`
	`print`
	`print(object)`
	`prompt`
	`prompt(query as string)`
	`range(max as int)`
	`range(begin as int, end as int)`
	`range(begin as int, end as int, step as int)`
	`reversed(enumerable)`
	`shell(filename, arguments as string)`
	`shellm(filename, arguments as (string) )`
	`shellp(filename, arguments as string)`
	`zip(first as IEnumerable, second as IEnumerable)`

Procedural Programming

Functions	`def sayHello():`

and Parameters	`def sayHello(name):`
	`def sayHello(name as string):`
	`def sayHello(*names):`
	`def sayHello(*names as (string)):`
	`def dobyref(ref x as int)`

Loops	`for i in range(0, 10): // from 0 to 9`
	`for i in [300, 100, 23, 1, 55]:`
	`for i in (1, 4, 98, 399, 1000, 34, 199):`
	`while i < 10:`
	`while not(i > 10):`

Exception Handling	`try:`
	`except e as ExceptionType:`
	`except e:`
	`ensure:`
	
	`raise "Some error"`

Closures	`f = def ():`
	`f = {cmd1(); cmd2()}`
	`double = { x | return 2*x}`

Generators	`<expression> for <declarations> in <iterator> if|unless <condition>`
	`[<expression> for <declarations> in <iterator> if|unless <condition>]`
	`yield b`

Types and Structures

Data types	{{}}
• Type Inference	{{}}
• Casting Types	{{}}
• Duck Typing	{{}}
• Callable Types	{{}}
• and Events	{{}}
• User-defined value types aka structs	{{}}
• Builtin Literals	{{}}
Lists And Arrays	{{}}
Hashtables	{{}}
Slicing	{{}}
Modules	{{}}

Object-oriented Features

Classes	{{}}
Fields And Properties	{{}}
Interfaces	{{}}
Operator overloading	{{}}
Enums	{{}}

Advanced Features

Macros	{{}}
Syntactic Macros	{{}}
Extensible Compiler Architecture	{{}}