

YARTBML





Yet Another Re-implementation of Thorsten Ball's Monkey Language

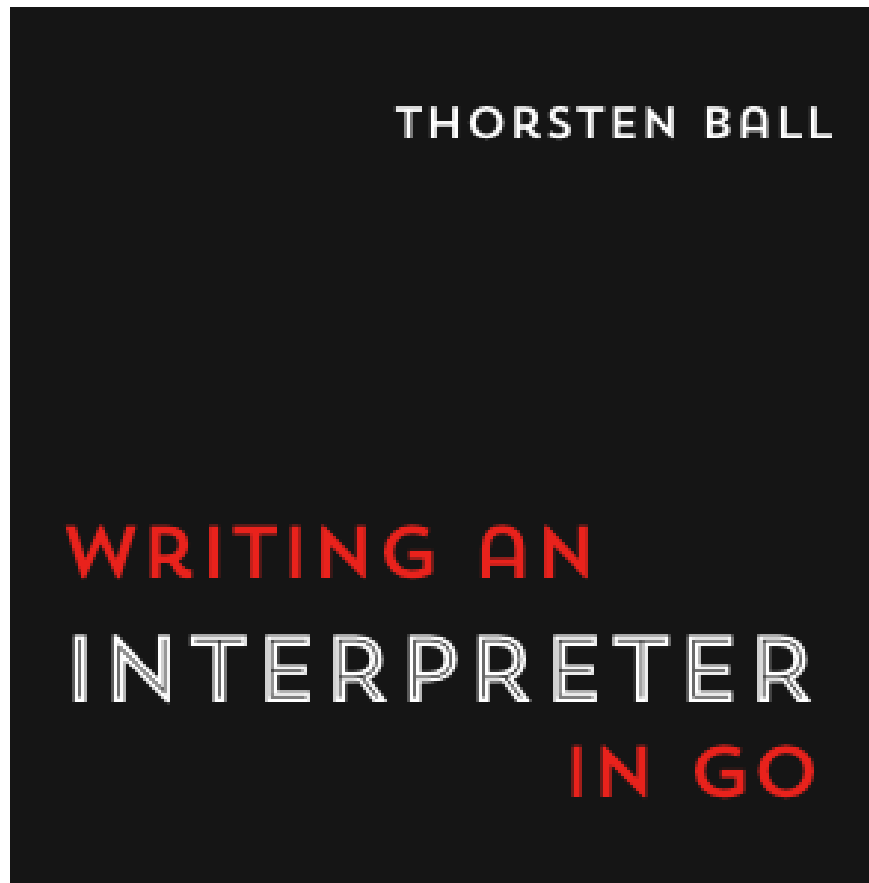
By: Dinesh Umasankar, Joseph Porrino, Katherine Banis, Paul Jensen



What is YARTBML?

Minimally Functional-Paradigm Inspired
Language (Based on SMoL)

-  Built on the foundation provided by Thorsten Ball's: "Writing an Interpreter in Go"
-  Inspired by the many forks of this foundation to provide the best feature set and experience
-  Focuses on the developer experience for building general-purpose applications
-  Learnable in a lunch break



Let's Prove It

File Format

- To start writing code in YARTBML, make a `.ybm1` file.
- All programs will be interpreted under UTF-8, and English Alphanumeric Characters only.
- Let's make an example fibonacci program.

Fibonacci Program

fibonacci.ybml

```
let fibonacci = fn(x) {  
  if (x == 0) {  
    return 0;  
  } else {  
    if (x == 1) {  
      return 1;  
    } else {  
      fibonacci(x - 1) + fibonacci(x - 2);  
    }  
  }  
};  
  
puts(fibonacci(10)) // Displays "55".
```

Core Language Principles

Parsing

Pratt Parsing Technique (form of Recursive Descent)

Interpreter / Evaluator

Tree-Walking Interpreter using the AST (Abstract Syntax Tree)

Data Types

- Integers: Whole numbers without a decimal component, e.g., `42` , `-7` .
- Booleans: Logical type representing `true` or `false` .
- Strings: A sequence of characters enclosed in double quotes, e.g., `"YARTBML is awesome!"` .
- Arrays: A list of elements, e.g., `[1, 2, 3, 4, "hello", true]` .
- Hashmaps: Key-value pairs, e.g., `{"name": "YARTBML", "isCool": true}` .

Data Types In Action

```
// Our team in an array  
let team = [dinesh, joseph, katherine, paul];  
let leader = team[0] // {"name": "Dinesh Umasankar", classification: "Senior"}
```

Functions

- First-Class Citizens
- Functions are a value-type
- Can be assigned to variables
- Passed as arguments
- Returned from other functions

```
let greet = fn(name) { return "Hello, " + name + "!"; };  
let message = greet("World");  
puts(message); // -> "Hello, World!"
```

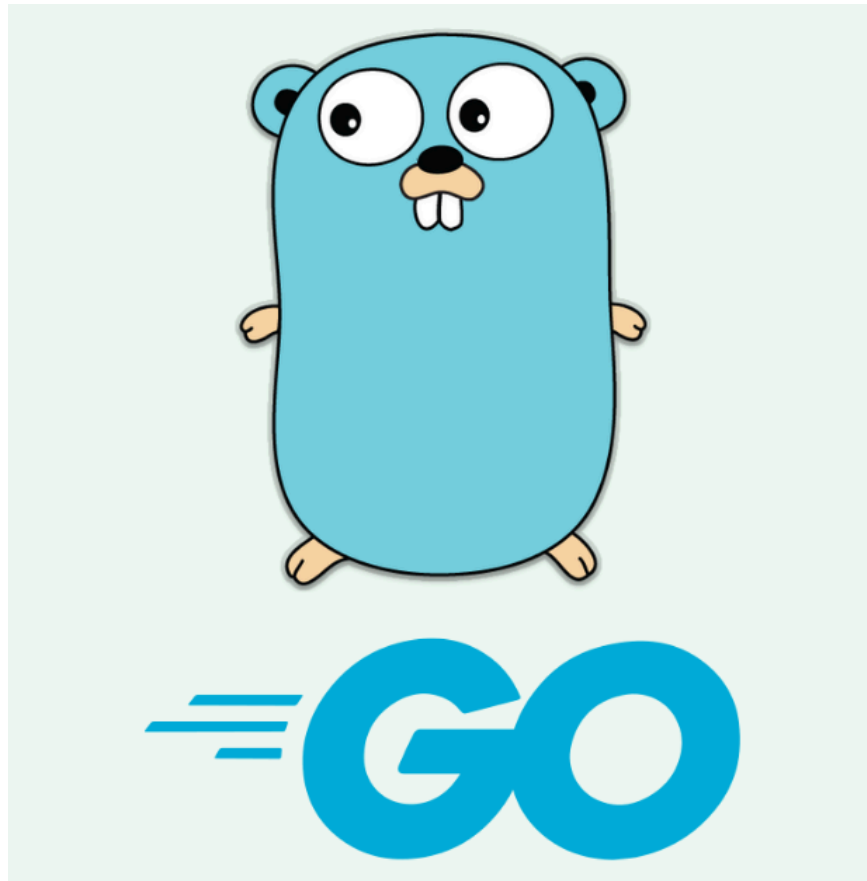
Operators

Traditional Arithmetic Operators w/ Precedence

- Equality-Expression: `==` or `!=`
 - Comparative Expression: `<` or `>`
 - Additive-Expression: `+` or `-`
 - Multiplicative-Expression: `*` or `/`
 - Prefix-Expression: `-` or `!`
-
- (FUTURE) *Operator Support for* `<=` and `>=`
 - (FUTURE) *Logical Operator Support:* `&&` and `||`

Memory Management Handled by Go Language

- Go's Runtime is statically linked into the interpreter binary, which contains a Garbage Collector.
- Interpreter is a binary file compiled to a specific machine architecture.



Built-In Functions

- `len` : gets length of characters in string or elements in array
- `first` : gets first element within array
- `last` : gets last element within array
- `rest` : gets rest of elements within array
- `push` : pushes an element at index 0 (*prepend*)
- `puts` : display object to terminal (*print*)

Language Development Tools

REPL

Read-Eval-Print Loop

Allows you to quickly test line-by-line code snippets in the terminal

CLI Build Tool

Executable Interpreter that takes in flags to interpret whole `.yaml` code files

Can be integrated with build-automation tools. *(CI/CD)*

Syntax Highlighting

Code with confidence by being able to read your code in colors which provide context.

Built for popular editors like: VSCode, Sublime*, NeoVim*

Team Responsibilities

Everyone is Responsible for:

- Testing: Done via Go's Testing Framework
- Documentation: Written in Markdown, Automated Release

✓ docs: YARTBML Programming Language Specification Document (#6)

Documentation Release #62: Commit [99c2928](#) pushed by dineshUmasankar

✓ Language Specification Document

Documentation Release #61: Pull request [#6](#) synchronize by dineshUmasankar

✓ Language Specification Document

Documentation Release #60: Pull request [#6](#) synchronize by dineshUmasankar

✓ docs: YARTBML Whitepaper (#10)

Documentation Release #59: Commit [0fb1026](#) pushed by dineshUmasankar

✓ YARTBML Whitepaper

Documentation Release #58: Pull request [#10](#) synchronize by dineshUmasankar

✓ YARTBML Whitepaper

Documentation Release #57: Pull request [#10](#) synchronize by pwjensen

✓ YARTBML Whitepaper

Documentation Release #56: Pull request [#10](#) opened by pwjensen

✓ Language Specification Document

Documentation Release #55: Pull request [#6](#) synchronize by BabyKangaroo117

Core Language

- Tokenizer & Lexer -> Joesph Porrino
- Parser & AST -> Katherine Banis
- Evaluator & Environment & Object & Built-In Functions -> Dinesh Umasankar

- **(FUTURE)** *Compile to Bytecode and Create Custom Stack-Based Virtual Machine*
- **(FUTURE)** *Compile to WASM & Build Browser-Based Coding Environment*

Automated Quality Assurance Tools

- Automated Testing Environment -> Dinesh Umasankar
- Automated Documentation -> Dinesh Umasankar
- Automated Interpreter Binary Release -> Dinesh Umasankar

Developer Experience

- Syntax Highlighter -> Paul Jensen
- **(FUTURE)** *Language Server Implementation (Autocomplete, Go-To Definition, In-Editor Documentation, etc.)*

Future Aspirations

- Support for logical `&&` and `||` Operators
- Support for `<=` and `>=` Operators
- Support for `%` (modulo) Operator
- Support for `++` and `--` Postfix Operators
- Support for Floating Point Operations
- Support for Macros
- Support for Loops
- Support for Regular Expressions

Future Aspirations

- Support for Import / Export Modules
- Language Server Protocol Implementation (Autocomplete, Go-To Definition, In-Editor Documentation, etc.)
- Improve Error Reporting
- Compile to WASM to allow for browser-based coding environment
- Compile the code into a bytecode definition and compute via Custom Stack-Based Virtual Machine
- Create Browser-Based Programming Tutorial

Future Aspirations

- Support for Concurrency (Requires Design Decision & Thoughts)
- Support for File I/O (Requires Design Decision & Thoughts)
- Investigation & Support for Type Systems
- Language Branding Page (About, Install, Documentation, Community, Development)

Thank You