Bang!

By Natalie Lau

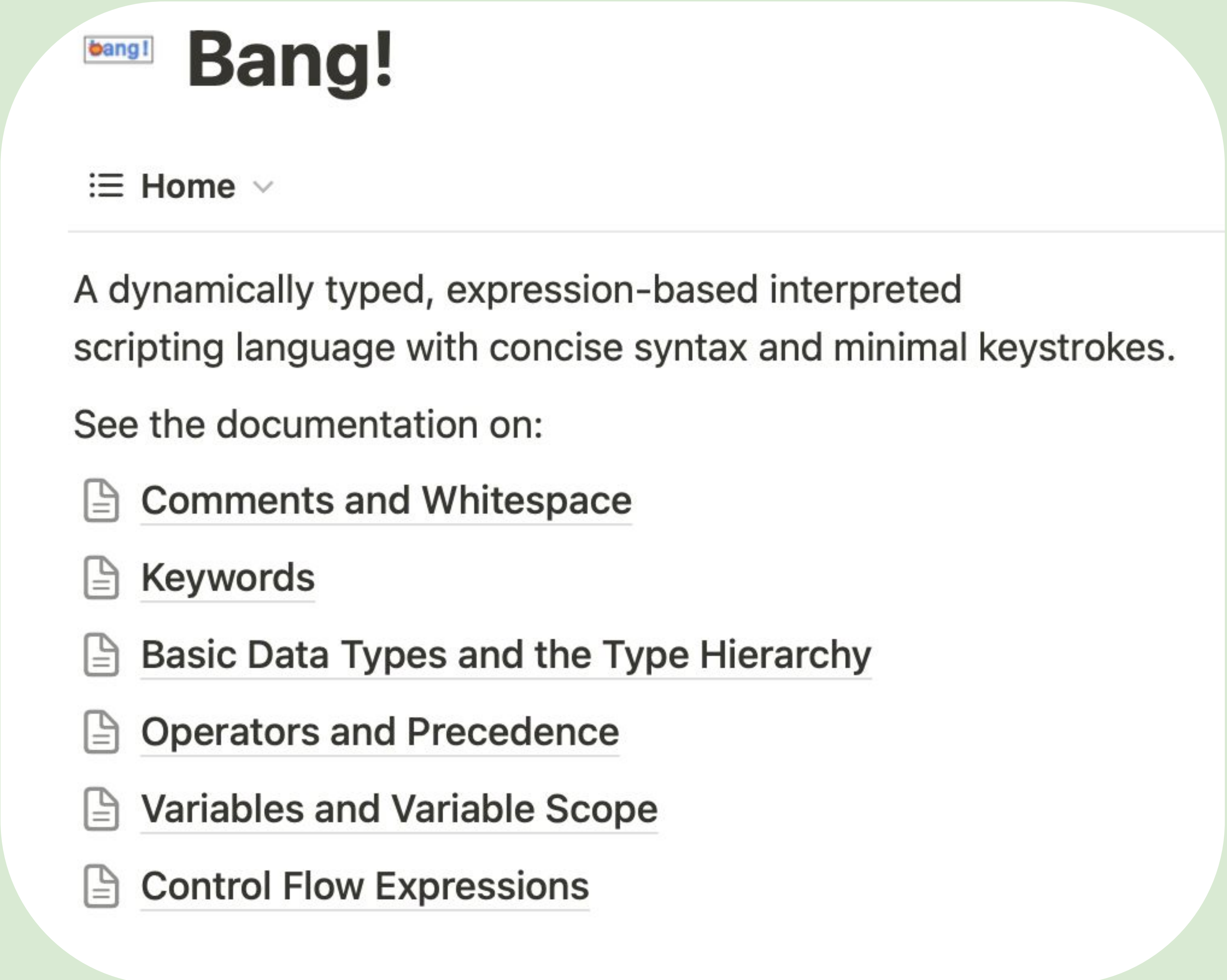
https://github.com/nklau/bang/

Motivation

Many enterprise languages aren't concise, and they often raise errors like null reference exceptions and type errors. My solution? Get rid of them!

Description

Bang! is a dynamically typed, expression-based interpreted scripting language with concise syntax and minimal keystrokes.



Bang!

Home

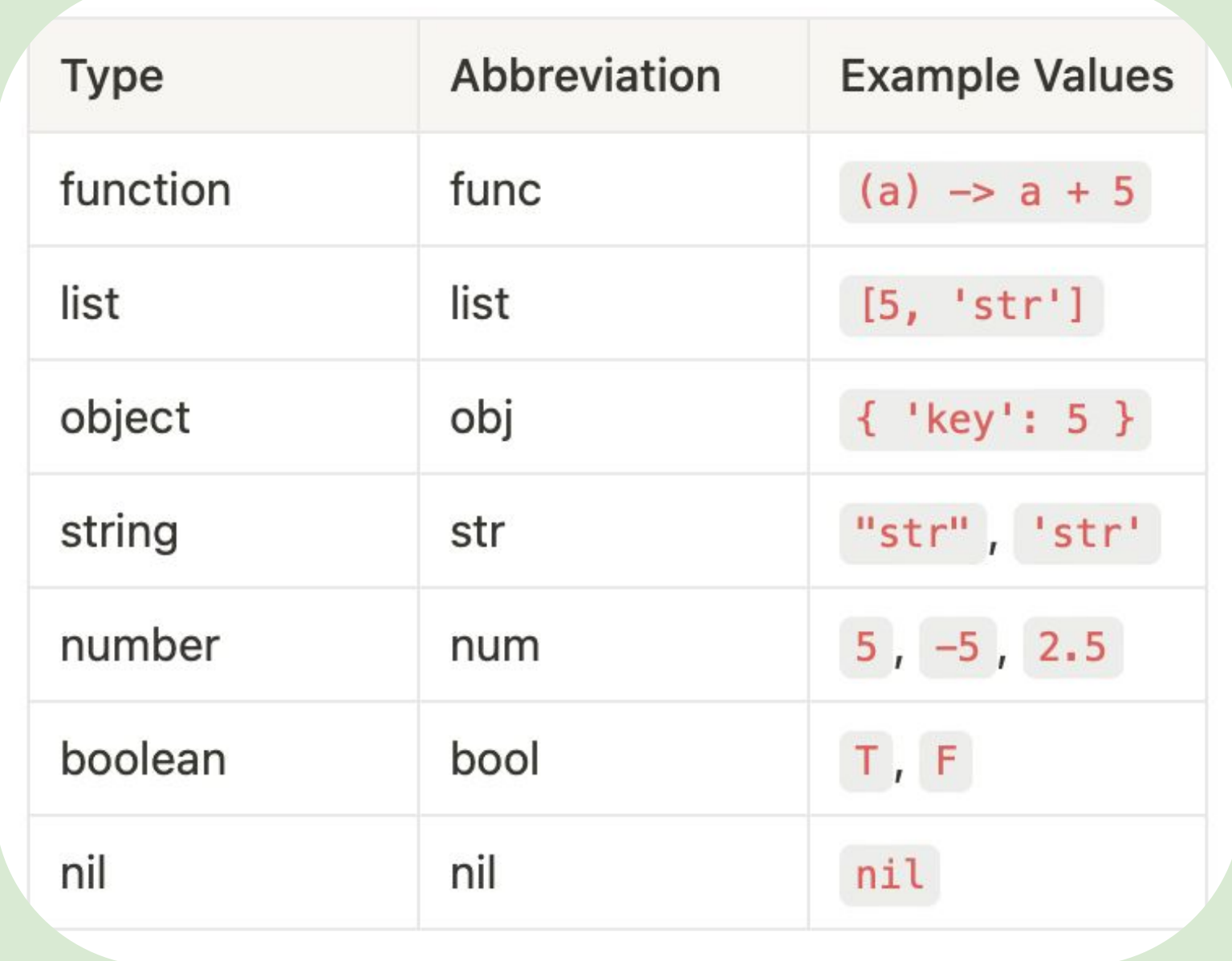
A dynamically typed, expression-based interpreted scripting language with concise syntax and minimal keystrokes.

See the documentation on:

- Comments and Whitespace
- Keywords
- Basic Data Types and the Type Hierarchy
- Operators and Precedence
- Variables and Variable Scope
- Control Flow Expressions

```
x = 5
y = (x %= 2)
prt($'x: {x}, y: {y}')
```

```
age = 20
age < 21 ? {
  prt("You're underage!")
} : {
  prt("You're of age!")
}
```



Type	Abbreviation	Example Values
function	func	`(a) -> a + 5`
list	list	`[5, 'str']`
object	obj	`{ 'key': 5 }`
string	str	`"str"`, `'str'`
number	num	`5`, `-5`, `2.5`
boolean	bool	`T`, `F`
nil	nil	`nil`

Challenges

Keeping the operators consistent across types was difficult because I wanted to avoid creating duplicate behaviors. Because there are no type errors, testing all possible combinations of operations isn't feasible, which made implementing and testing operators one of the most challenging parts of this project.

Up Next

This project will be continued next semester. The next step is the interpreter, which involves semantic analysis and producing standard output. Additionally, although Bang! already has langauge documentation on Notion, I plan to make a website for developers to practice their code golfing skills in Bang!

```
add = (a, b) -> {
  sum = a + b
  sum
}
```

```
age = 20
ofAge = age < 21 ? "You're underage!"
ofAge
```

```
[1, 2].lp((elem, index) -> {
  prt($'{elem} at index {index}')
})
```

```
5.lp {
  prt($0)
  $0 == 3 ? brk
}
```

- Typescript - Notion -