

A blue parallelogram and a light green parallelogram are positioned in the upper-left corner of the slide. The blue shape is partially behind the green one. Both shapes are oriented diagonally, matching the overall geometric theme of the background.

Bang!

Natalie Lau
4/10/24



Project Description

- Dynamic, expression-based language
- Concise syntax
- No type errors
- No null reference exceptions

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



Project Goals

- Ideal for code-golfing
- Unique perspective on data types
- Teaches devs to trace programs

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



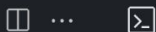
Project Justification

- Learn about language design
- Better understanding of components
- Pushing semantic boundaries

```
' Hello world ' / nil + 0
```



≡ interpreterExample.bang ×



bash + ▾ □ □ ... ×

programs > ≡ interpreterExample.bang

```

1  prt("'5 + 3" outputs:', 5 + 3)
2  prt("'3 - 5" outputs:', 3 - 5)
3  prt("'5 + T" outputs:', 5 + T)
4  prt("'5 - T" outputs:', 5 - T)
5  prt("'T + 6" outputs:', T + 6)
6  prt("'T - 6" outputs:', T - 6)
7  prt("'5 + F" outputs:', 5 + F)
8  prt("'5 - F" outputs:', 5 - F)
9  prt("'F + 5" outputs:', F + 5)
10 prt("'F - 5" outputs:', F - 5)
11 prt("'5 + nil" outputs:', 5 + nil)
12 prt("'5 - nil" outputs:', 5 - nil)
13 prt("'nil + 5" outputs:', nil + 5)
14 prt("'nil - 5" outputs:', nil - 5)
15 prt("'T + T" outputs:', T + T)
16 prt("'T + F" outputs:', T + F)
17 prt("'F + T" outputs:', F + T)
18 prt("'F + F" outputs:', F + F)
19 prt("'T + nil" outputs:', T + nil)
20 prt("'nil + T" outputs:', nil + T)
21 prt("'F + nil" outputs:', F + nil)
22 prt("'nil + F" outputs:', nil + F)
23 prt("'T - T" outputs:', T - T)
24 prt("'T - F" outputs:', T - F)
25 prt("'F - T" outputs:', F - T)
26 prt("'F - F" outputs:', F - F)
27 prt("'T - nil" outputs:', T - nil)
28 prt("'nil - T" outputs:', nil - T)

```



The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit <https://support.apple.com/kb/HT208050>.

```

MacBook-Pro-5:bang natalielau$ ts-node src/bang.ts programs/interpreterExample.bang run
Interpreter output:
"5 + 3" outputs: 8
"3 - 5" outputs: -2
"5 + T" outputs: 6
"5 - T" outputs: 4
"T + 6" outputs: 7
"T - 6" outputs: -5
"5 + F" outputs: 5
"5 - F" outputs: 5
"F + 5" outputs: 5
"F - 5" outputs: -5
"5 + nil" outputs: 5
"5 - nil" outputs: 5
"nil + 5" outputs: 5
"nil - 5" outputs: -5
"T + T" outputs: T
"T + F" outputs: T
"F + T" outputs: T
"F + F" outputs: F
"T + nil" outputs: T
"nil + T" outputs: T
"F + nil" outputs: F
"nil + F" outputs: F
"T - T" outputs: F
"T - F" outputs: T
"F - T" outputs: T
"F - F" outputs: F
"T - nil" outputs: T
"nil - T" outputs: F

```












☰ 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](#)
-  [Implicit Variable Declarations](#)
-  [Control Flow Expressions](#)

Project Demo





Project Challenges

- Context-sensitive grammar
 - Repeat operators like ":"
- Consistency in language design

```
season = 'fall'
mtch season {
  cs 'spring': prt("It's springtime!")
  cs 'summer': prt("It's summertime!")
  cs 'fall', 'winter': prt("It's cold!")
}
```




Current Project Status

- Language documentation complete
- Lexer complete
- Parser complete
- Part of interpreter complete
 - Runs source code
 - Produces output

Questions?

