Building our own JavaScript

How do interpreted languages work?

Bruno Paulino

Bruno Paulino

Tech Lead at N26 | Platform Engineering

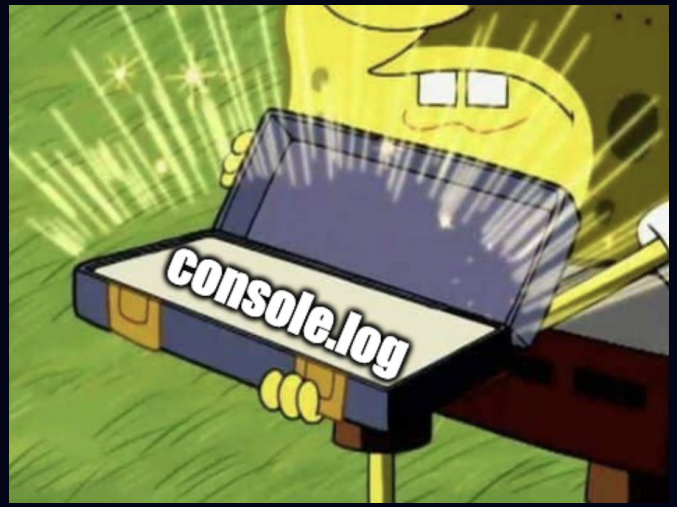
Software Engineer
|> originally from Brazil 🇧🇷
|> based in Vienna 🇦🇹

Big fan of Programming Languages and Web Tech.

brunojppb @bpaulino0

Did you know?

The console object isn't part of JavaScript 🤯



Agenda

- 📄 It all starts from plain text
  - Side quest on JavaScript engines and runtimes
- 🪙 Tokens, Lexers/Tokenizers and why they are important
- 📦 Parsers and Abstract Syntax Trees (ASTs)
- 🧵 Interpreters and where the action really happens
- 👩‍💻 Live demo of JavaScript expression evaluator

It all starts from plain text

app.js JavaScript

```
let result = 1 + 1;
```

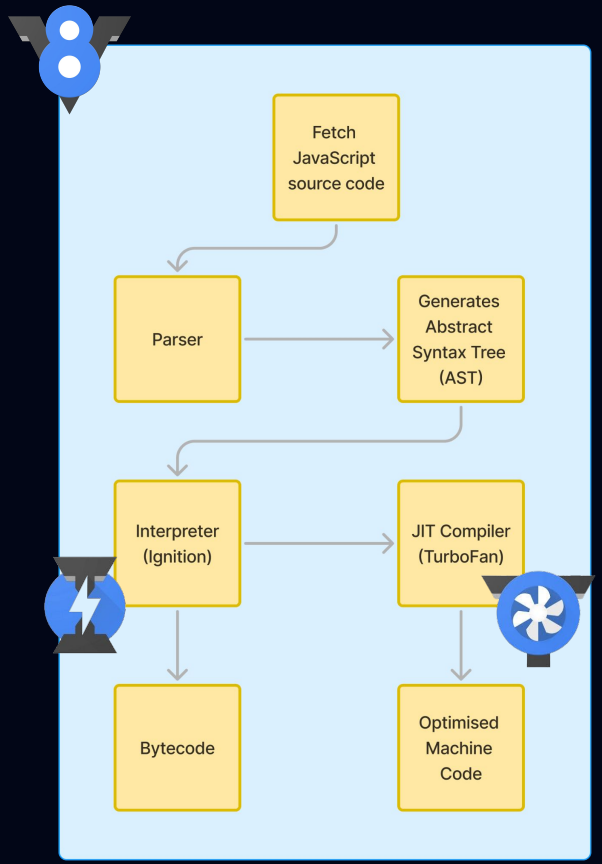
It all starts from plain text

app.js JavaScript

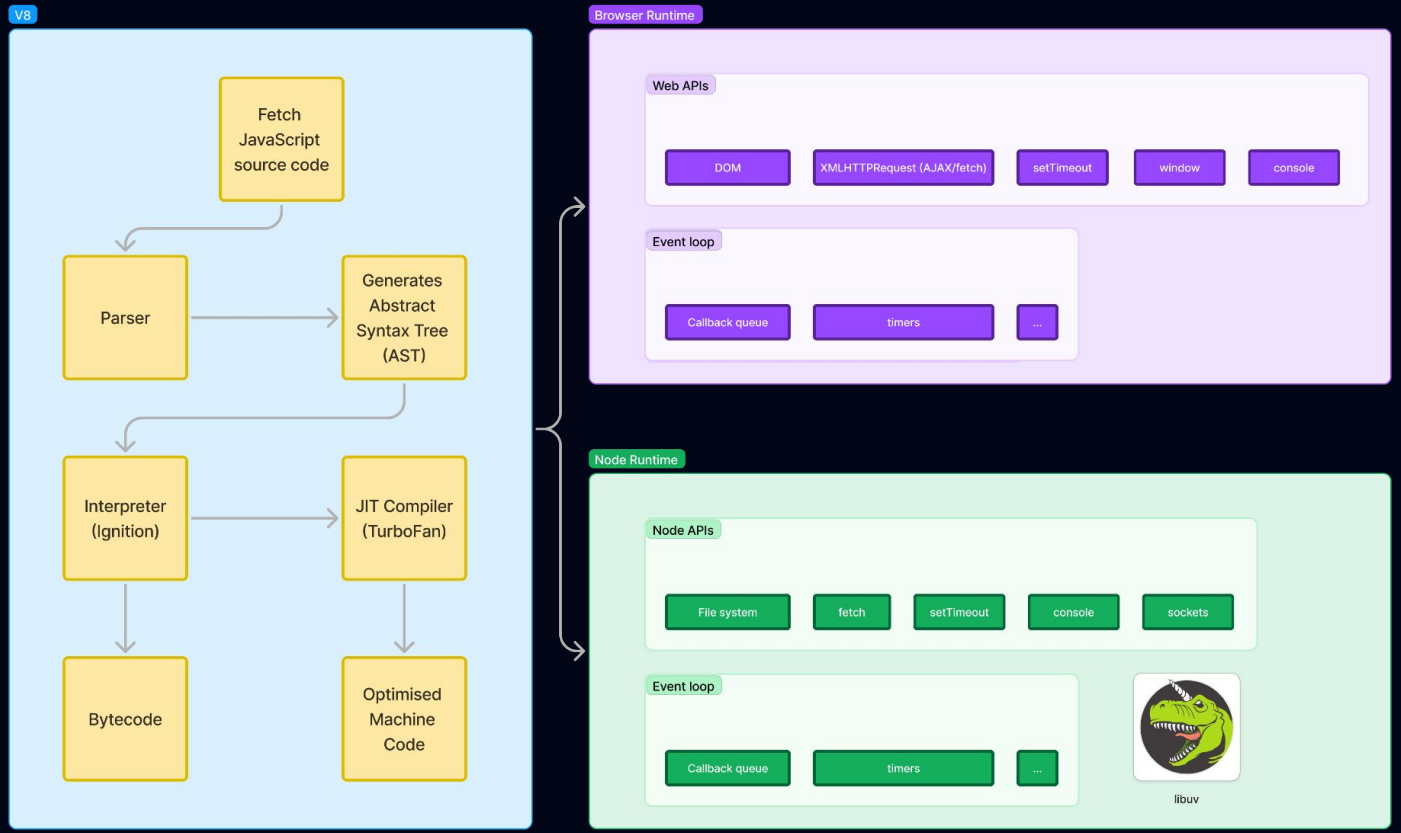
```
let result = 1 + 1;
```

Let's see what Chrome does

JavaScript engine takes over



JavaScript engine and runtimes



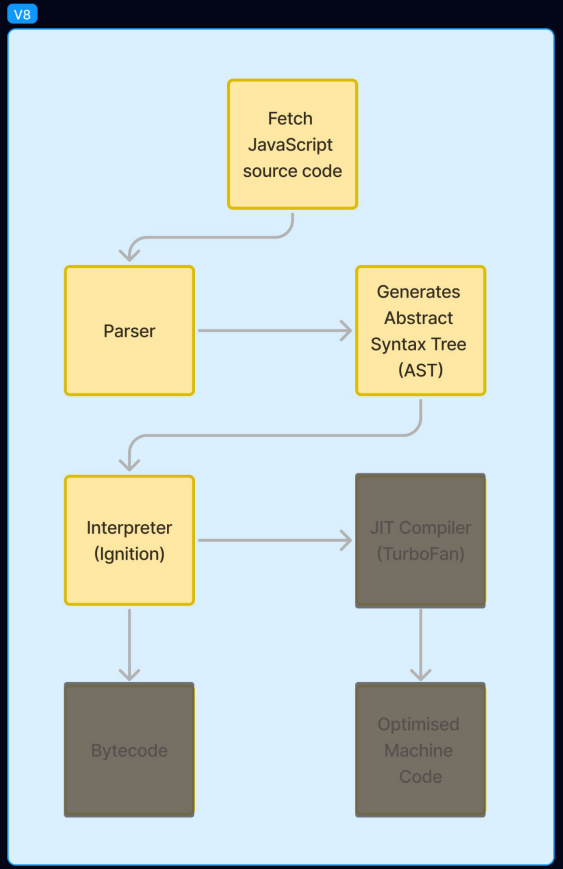
Companies trying to improve runtimes

Bun

Deno

Workerd

Coming back to our main quest



Lexers and why they are important

app.js JavaScript

```
let result = 1 + 1;
```

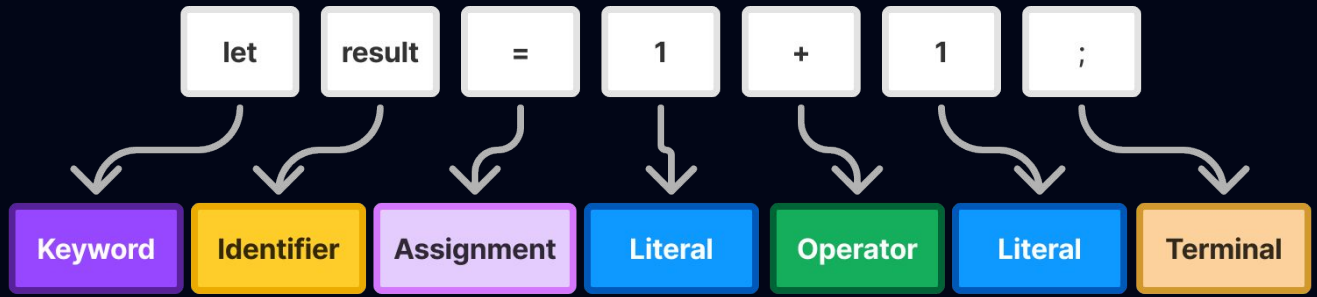
Lexers and why they are important

Lexical analysis

- Location info
- Literal values
- No syntax context

app.js JavaScript

```
let result = 1 + 1;
```



Parsers, Abstract Syntax Trees (ASTs) and how the magic happens

app.js JavaScript

```
1 + 3 * 2 + 1
```

app.js JavaScript

```
1 + 6 + 1
```

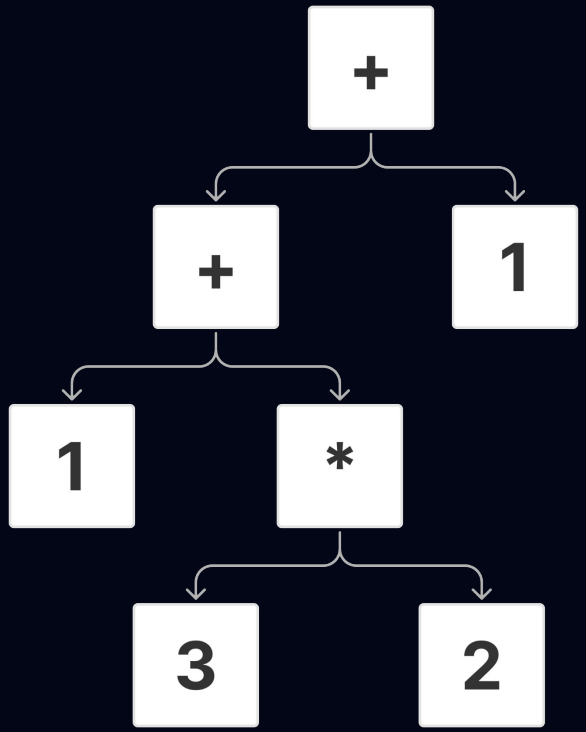
app.js JavaScript

```
8
```

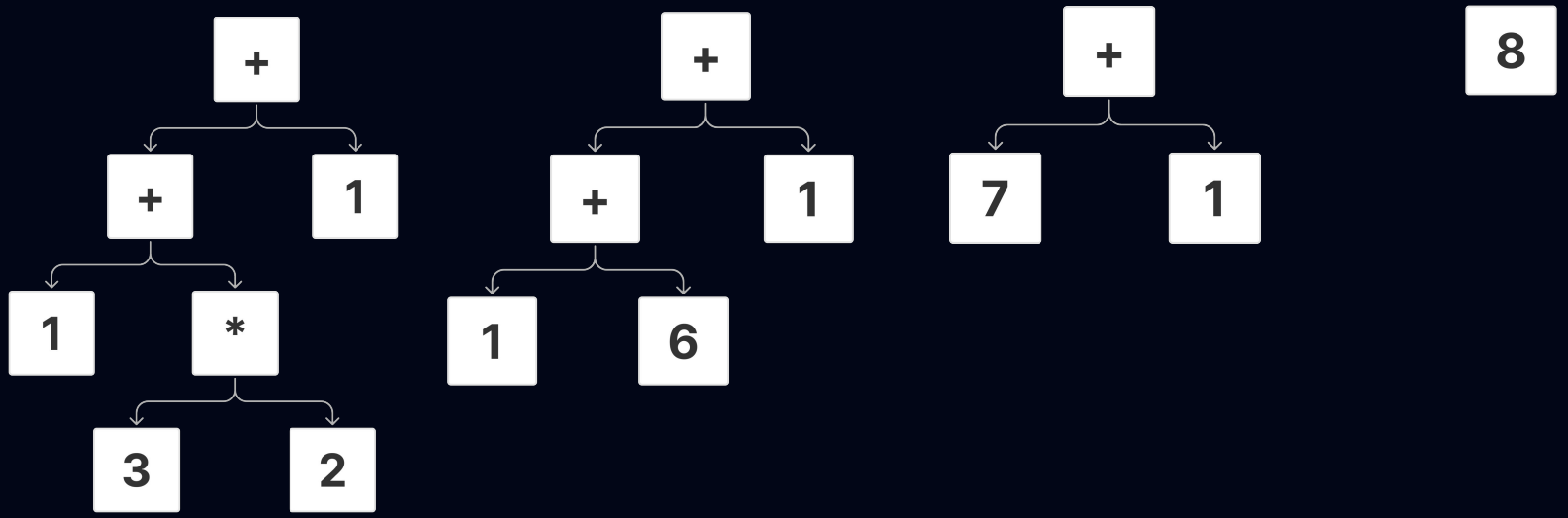
Parsers, Abstract Syntax Trees (ASTs) and how the magic happens

app.js JavaScript

```
1 + (3 * 2) + 1
```



Parsers, Abstract Syntax Trees (ASTs) and how the magic happens



Formal Grammars

> A formal grammar describes **which strings from an alphabet of a formal language are valid according to the language's syntax**. A grammar does not describe the meaning of the strings or what can be done with them in whatever context — only their form. A formal grammar is defined as a **set of production rules** for such strings in a formal language.

Source: wikipedia

Formal Grammars

Given a finite set of rules, you can derive an infinite number of strings

- These rules are called productions
  - Composed by a head (its name) and a body (description of what it generates)
  - Bodies are composed by symbols called terminals and nonterminals

Parsers, Abstract Syntax Trees (ASTs) and how the magic happens

```
1  expression → term ;
2
3  term       → factor ( ( "-" | "+" ) factor )* ;
4  factor     → primary ( ( "/" | "*" ) primary )* ;
5  primary    → NUMBER | "(" expression ")" ;
```

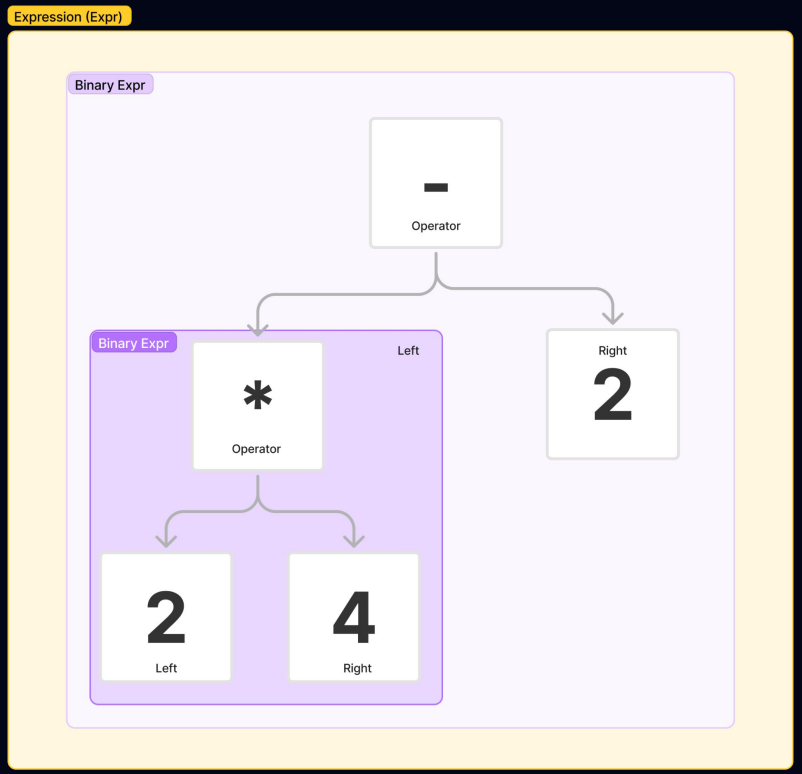
This is called a Recursive Descent Parser

> it starts from the top outermost grammar rule and works its way down into the nested subexpressions before finally reaching the leaves of the syntax tree.

Further read: ECMASCript grammar

Source of the quote: Crafting Interpreters

Exploring the AST



AST Explorer | Snippet | JavaScript | acorn | Transform | default | ?

Parser: acorn-8.7.0

```
1 2 * 4 - 2
```

Tree | JSON | 2ms

☑ Autofocus ☑ Hide methods ☑ Hide empty keys ☑ Hide location data ☑ Hide type keys

```
- Program {
  - body: [
    - ExpressionStatement {
      - expression: BinaryExpression {
        - left: BinaryExpression {
          - left: Literal {
              value: 2
              raw: "2"
            }
            operator: "*"
          - right: Literal = $node {
              value: 4
              raw: "4"
            }
          }
          operator: "-"
        - right: Literal {
            value: 2
            raw: "2"
          }
        }
      }
    ]
    sourceType: "module"
  }
```

Built with React, Babel, Font Awesome, CodeMirror, Express, and webpack | GitHub | Build: 8888701

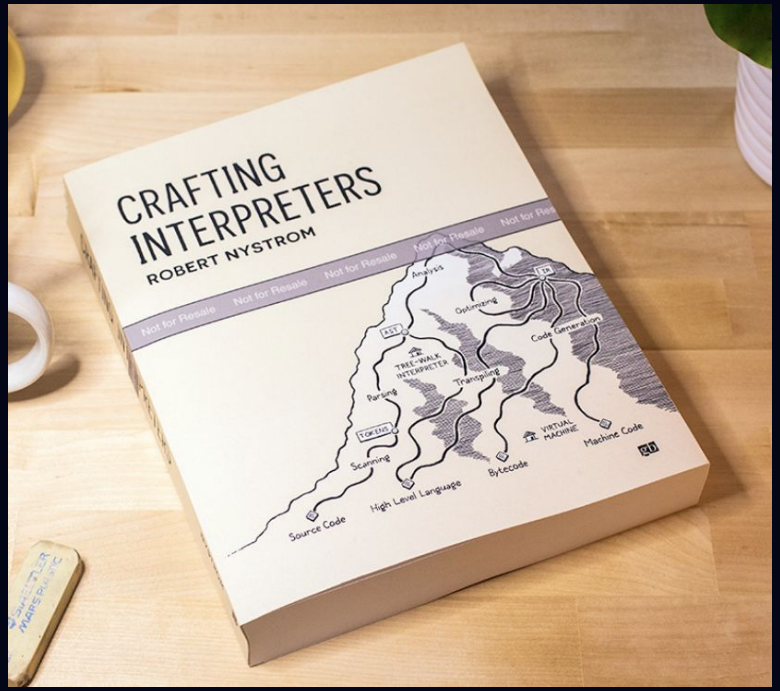
Interpreters and where the action really happens

The interpreter takes the AST as input and walks through it recursively and evaluates each node.

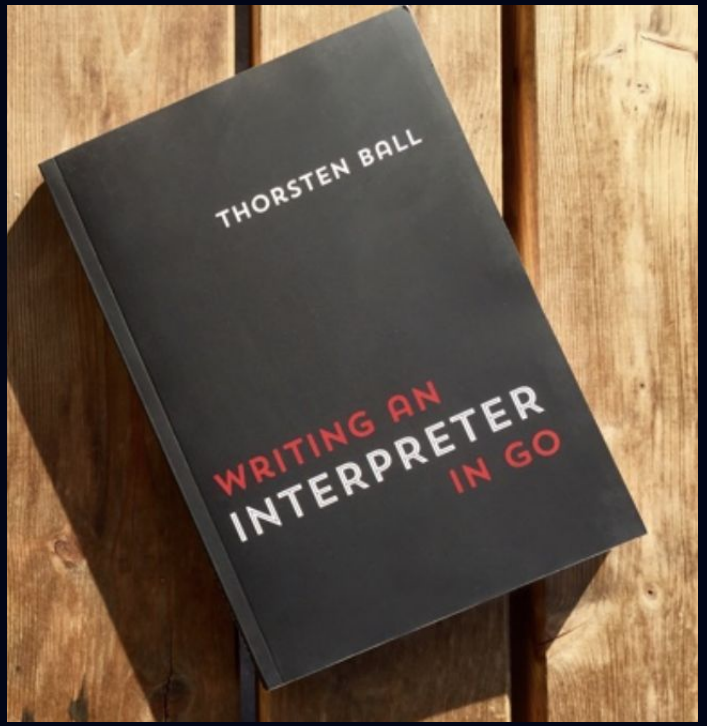
The Interpreter bridges your own language with the host language through the runtime. In Node is C++, in Deno is Rust and in Bun is Zig.

This is called a tree-walker interpreter

Further reading



[Crafting interpreters]



[Writing an Interpreter in Go]

Demo time

github.com/brunojppb/building-our-own-js