Andrei Lisnic
CTO @ Salt Edge

https://keybase.io/andrei

https://saltedge.com/jobs

Writing a programming language in Ruby

What is a programming language

- set of rules (syntax)
- execution (compiled, interpreted, etc)

Set of rules - syntax

```
$ ruby -e 'puts("OHAI'
-e:1: unterminated string meets end of file
-e:1: syntax error, unexpected end-of-input, expecting ')'
```

Set of rules - syntax

```
$ ruby -e 'puts("OHAI'
-e:1: unterminated string meets end of file
-e:1: syntax error, unexpected end-of-input, expecting ')'

$ ruby -e 'puts("OHAI")'
OHAI
```

Steps

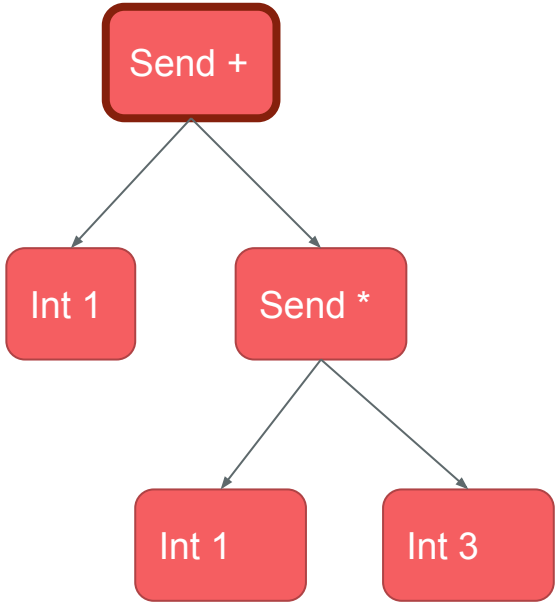
1. Define rules
2. ???

Abstract Syntax Tree

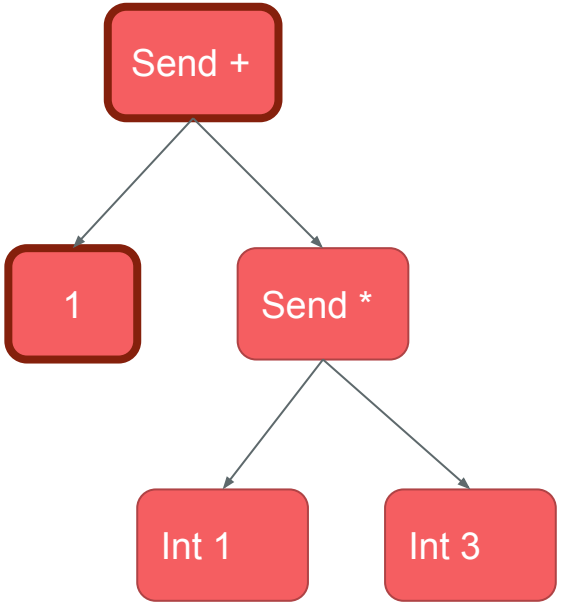
In computer science, an **abstract syntax tree** (**AST**), or just **syntax tree**, is a **tree** representation of the **abstract syntactic** structure of source code written in a programming language. Each node of the **tree** denotes a construct occurring in the source code.

```
pry(main)> Parser::CurrentRuby.parse("1 + (1 * 3)")
=> s(:send,
  s(:int, 1), :+,
  s(:begin,
    s(:send,
      s(:int, 1), :*,
      s(:int, 3))))
```

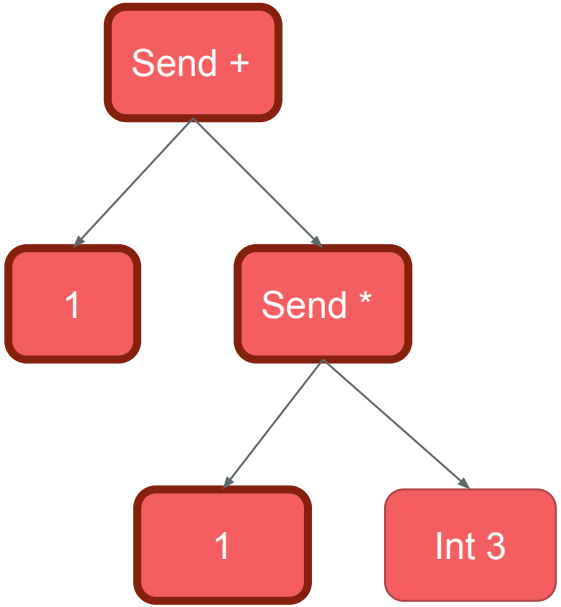
```
1 + (1 * 3)
```



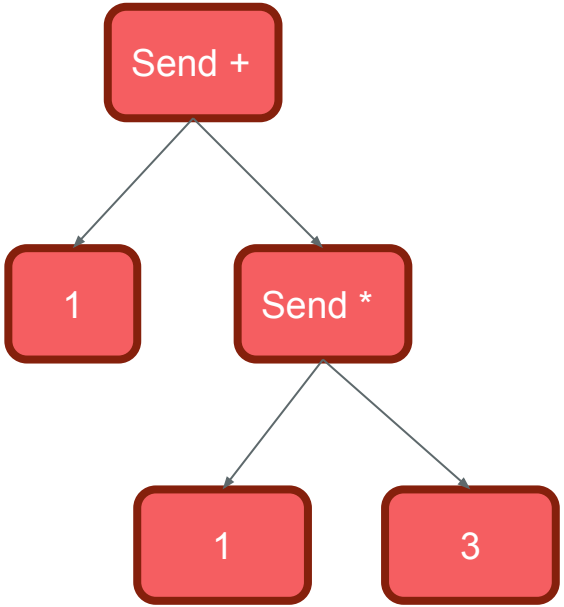
```
1 + (1 * 3)
```



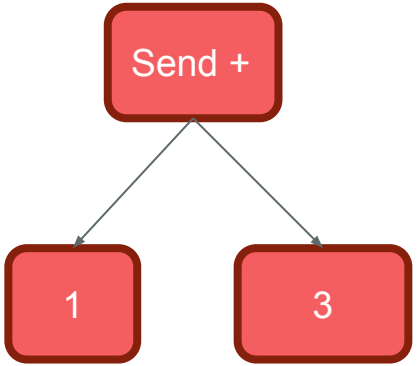
```
1 + (1 * 3)
```



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



1 + 3



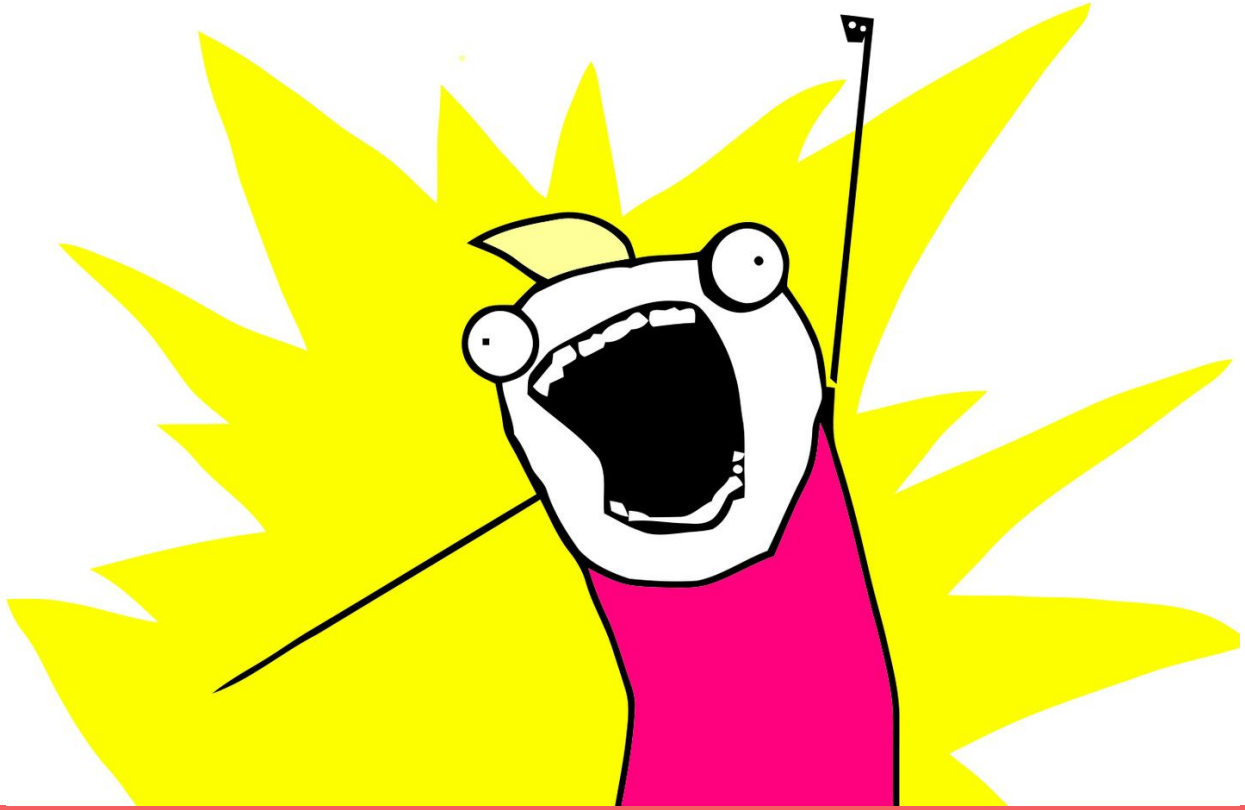
4

4

Steps

1. Define rules
2. Build AST
3. Evaluate AST

Let’s write a programming language!



```
(puts "Please input your name")
(set name (gets))
(puts (+ "hello " name))
```

LISP

Steps

1. **Define rules**
2. Build AST
3. Evaluate AST

PEG - Parsing expression grammar

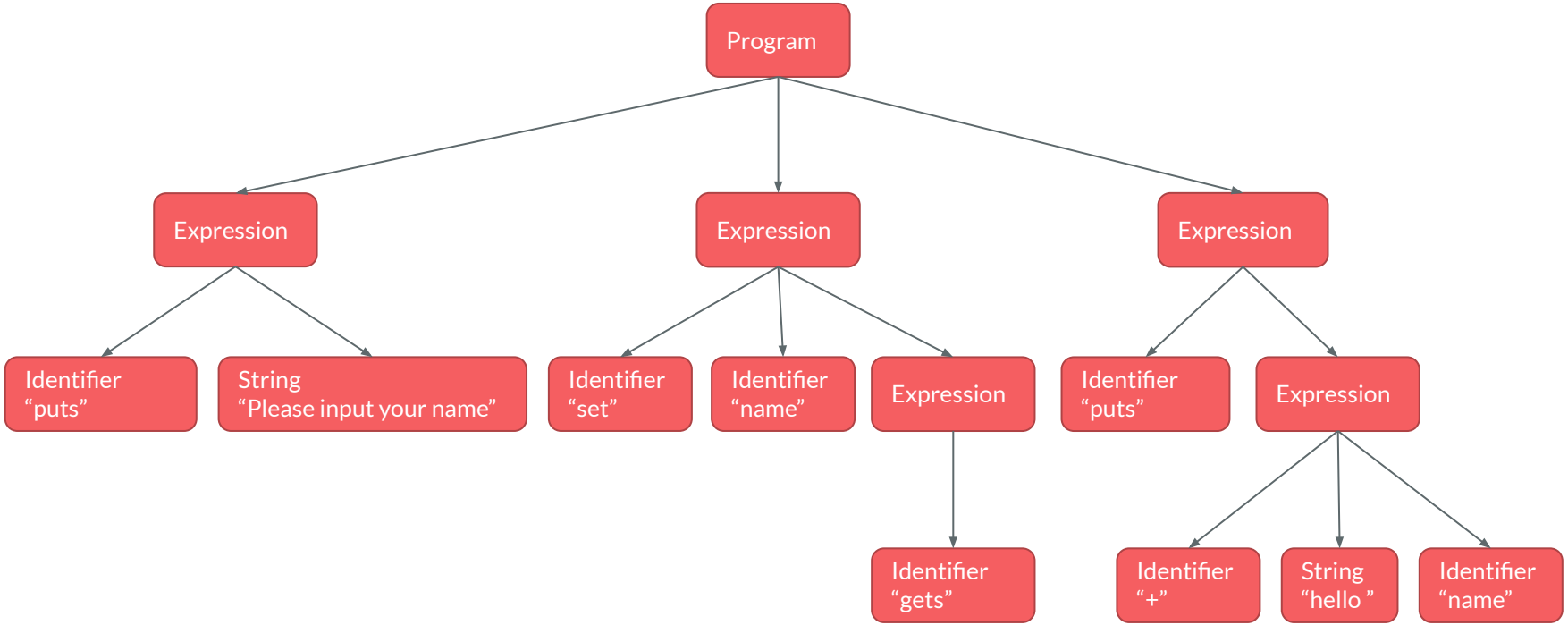
… is a type of analytic formal grammar, i.e. it describes a formal language in terms of a set of rules for recognizing strings in the language.

Let’s code!

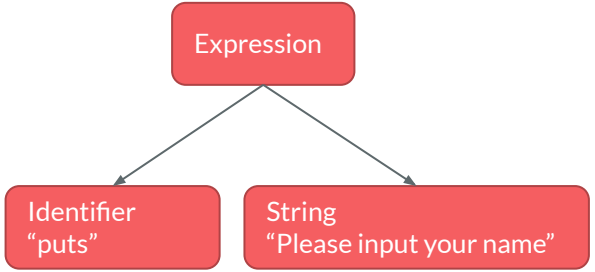
Steps

1. ~~Define rules~~
2. Build AST
3. Evaluate AST

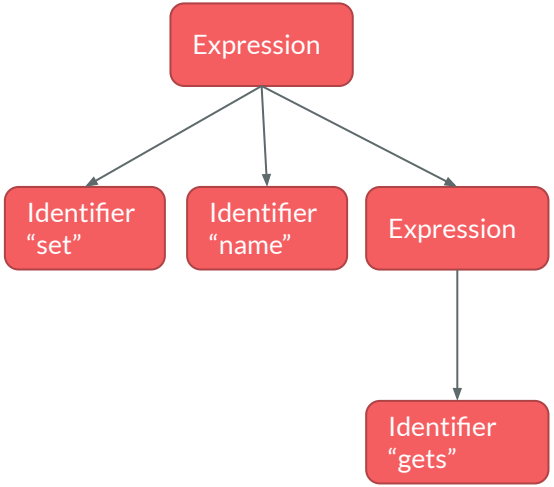
```
(puts "Please input your name")
(set name (gets))
(puts (+ "hello " name))
```


Program
Expression
Expression
Expression
Identifier
“puts”
String
“Please input your name”
Identifier
“set”
Identifier
“name”
Expression
Identifier
“puts”
Expression
Identifier
“gets”
Identifier
“+”
String
“hello ”
Identifier
“name”

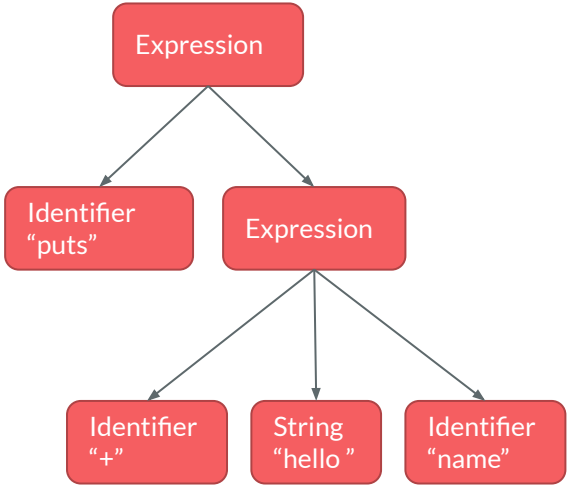
```
(puts "Please input your name")
```

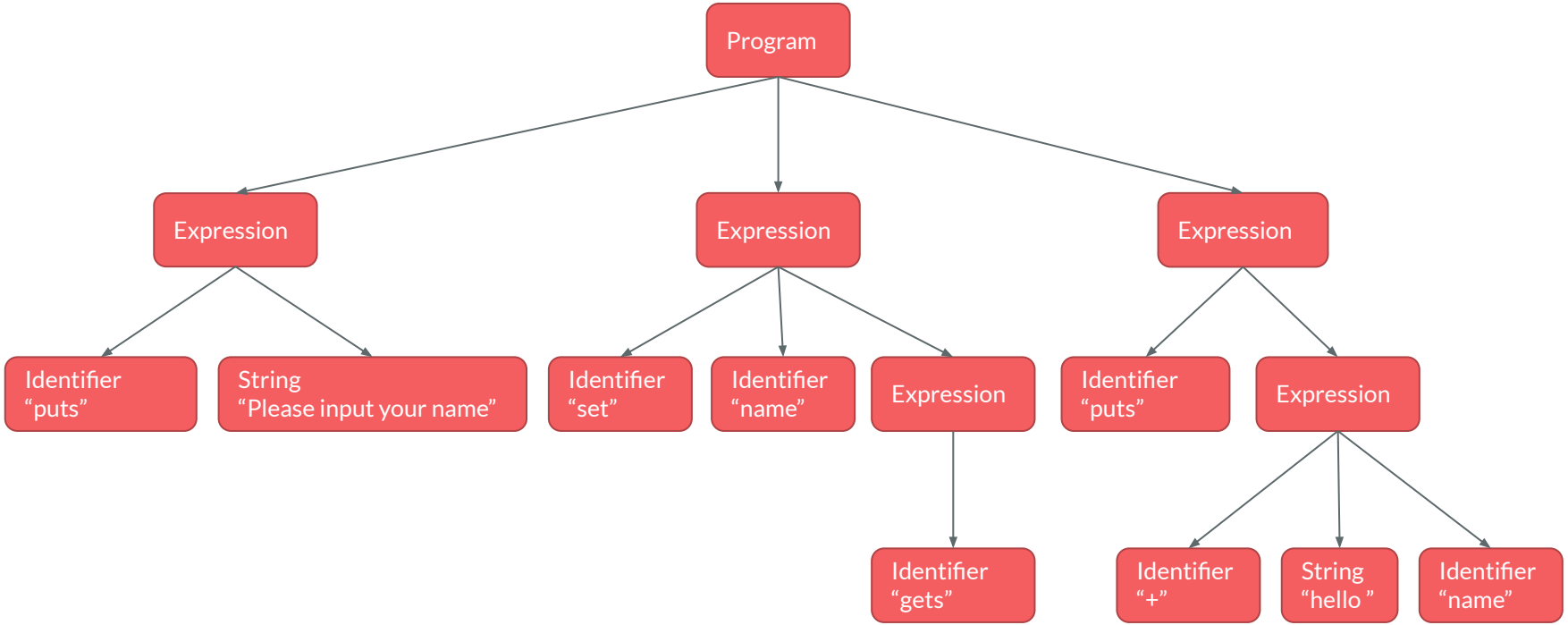


```
(set name (gets))
```



```
(puts (+ "hello " name))
```




Program
Expression
Expression
Expression
Identifier
“puts”
String
“Please input your name”
Identifier
“set”
Identifier
“name”
Expression
Identifier
“puts”
Expression
Identifier
“gets”
Identifier
“+”
String
“hello ”
Identifier
“name”

Let’s code!

Steps

1. ~~Define rules~~
2. ~~Build AST~~
3. Evaluate AST

How to draw an owl

1.

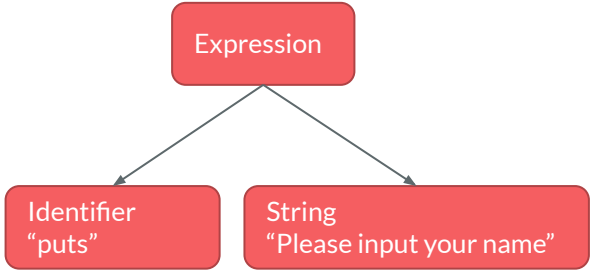
2.

Let’s code!

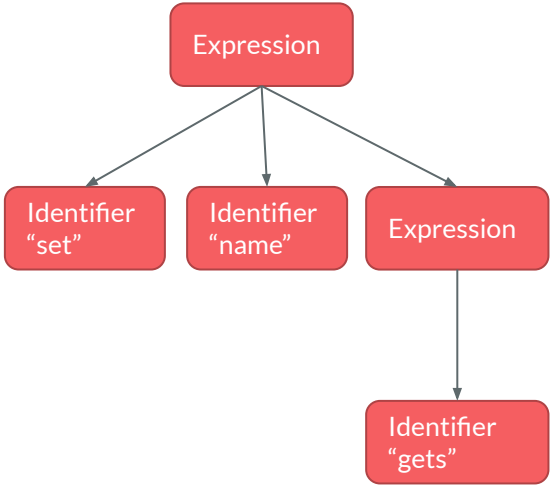
1. Draw some circles

2. Draw the rest of the f ing owl

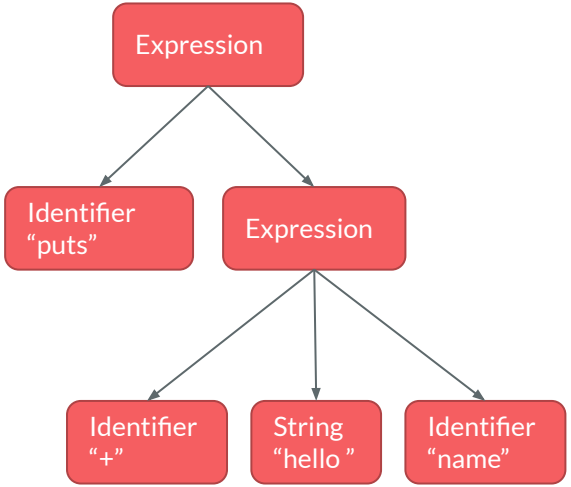
```
(puts "Please input your name")
```



```
(set name (gets))
```



```
(puts (+ "hello " name))
```



Steps

1. ~~Define rules~~
2. ~~Build AST~~
3. ~~Write implementation line by line~~

Questions?

https://keybase.io/andrei

https://saltedge.com/jobs