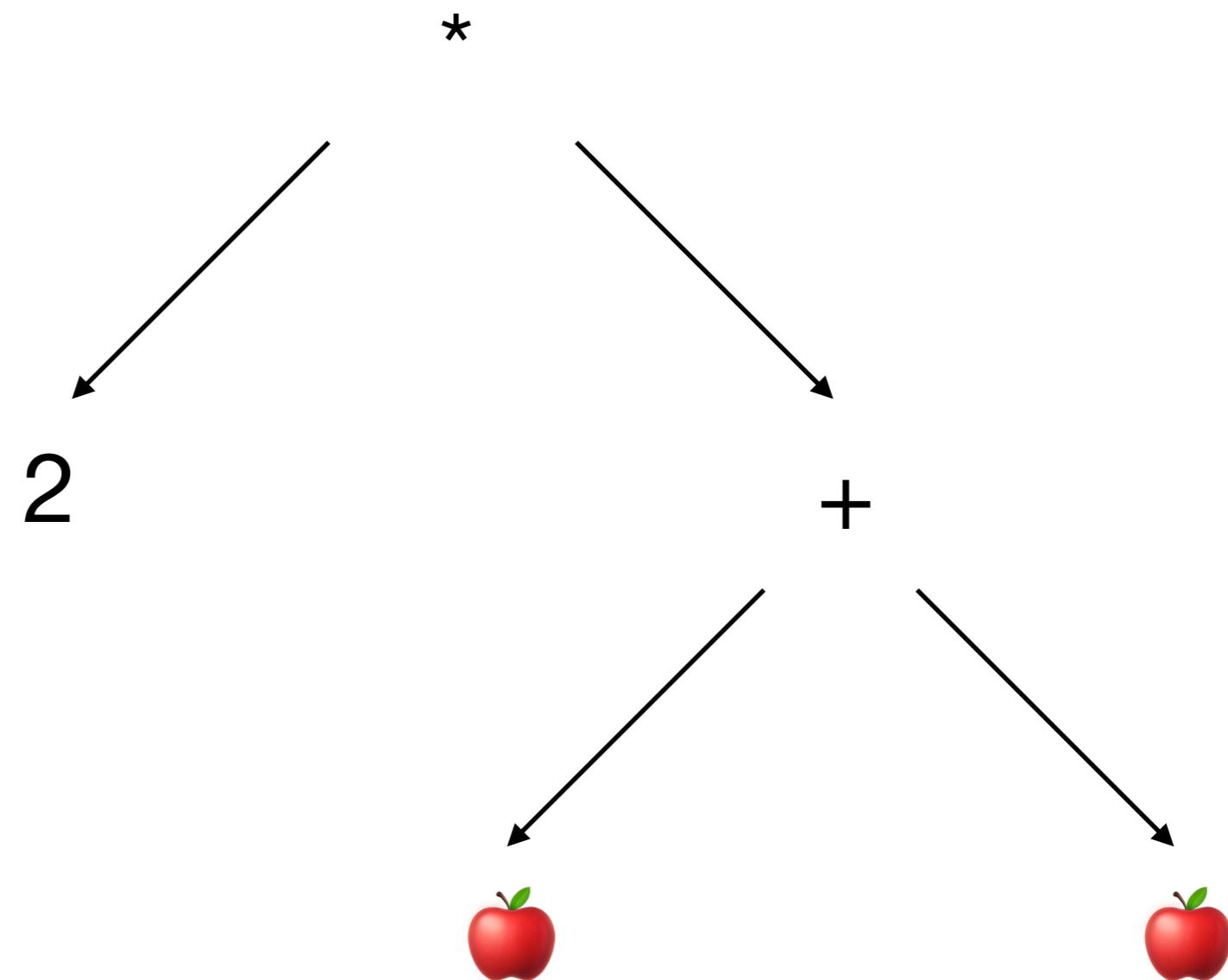


# Language Parsing in Action

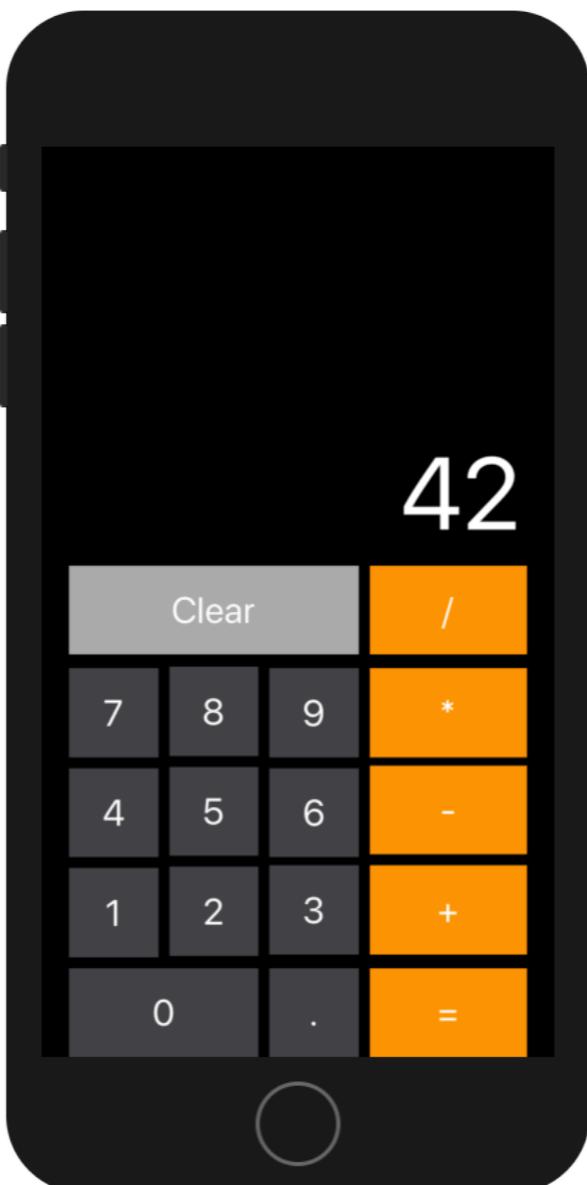
2( + 

# Syntax Tree



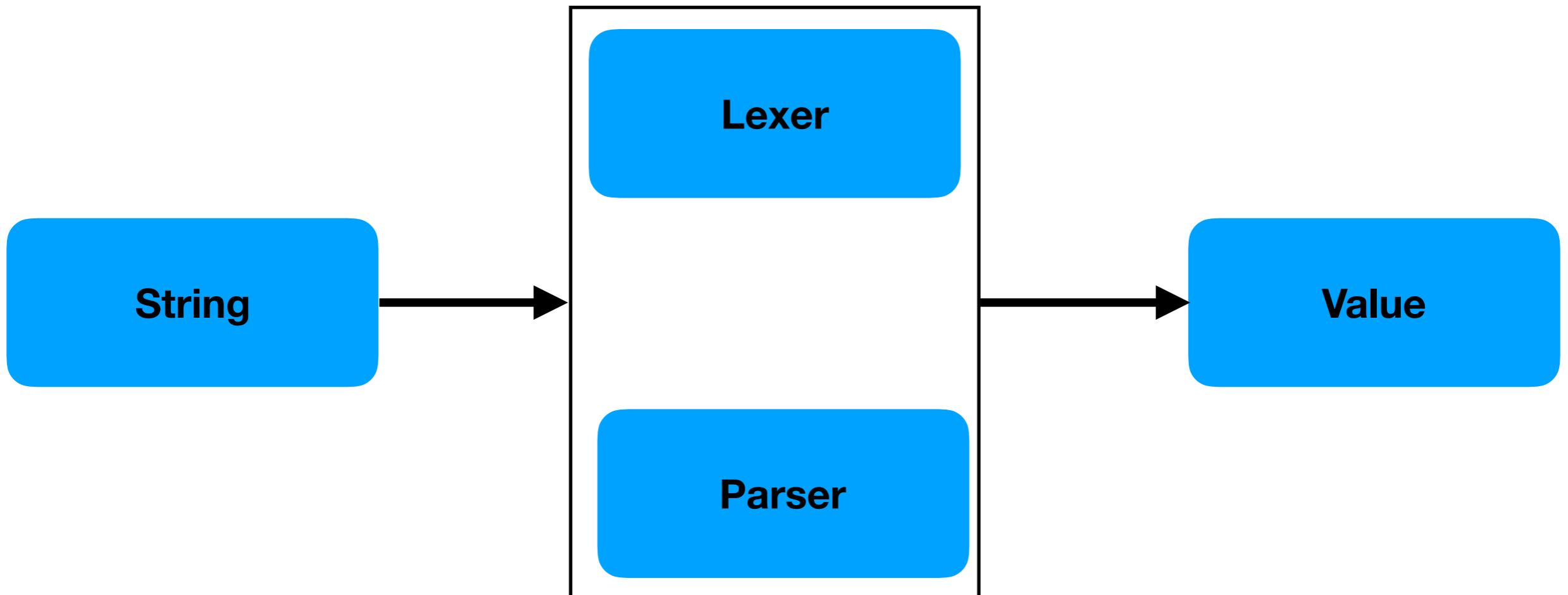
# Demo

How to build a calculator ...



... without using NSExpression

# So, how do we write our own lexer and parser?



We don't need to - We can use a parser generator

# Grammar - Our set of rules

**Number:**

Any combination of string

„0“...“9“

**Expression:**

Number

Expression + Expression

Expression - Expression

Expression \* Expression

Expression / Expression

# Parser Generator

- Yacc - Yet another Compiler-Compiler

```
expr : number { $$ = $1; }
| expr '*' expr { $$ = $1 * $3; }
| expr '/' expr { $$ = $1 / $3; }
| expr '+' expr { $$ = $1 + $3; }
| expr '-' expr { $$ = $1 - $3; }

number : INTEGER { $$ = $1; }
| FLOAT { $$ = $1; }
```

- Lex - lexical analyzer

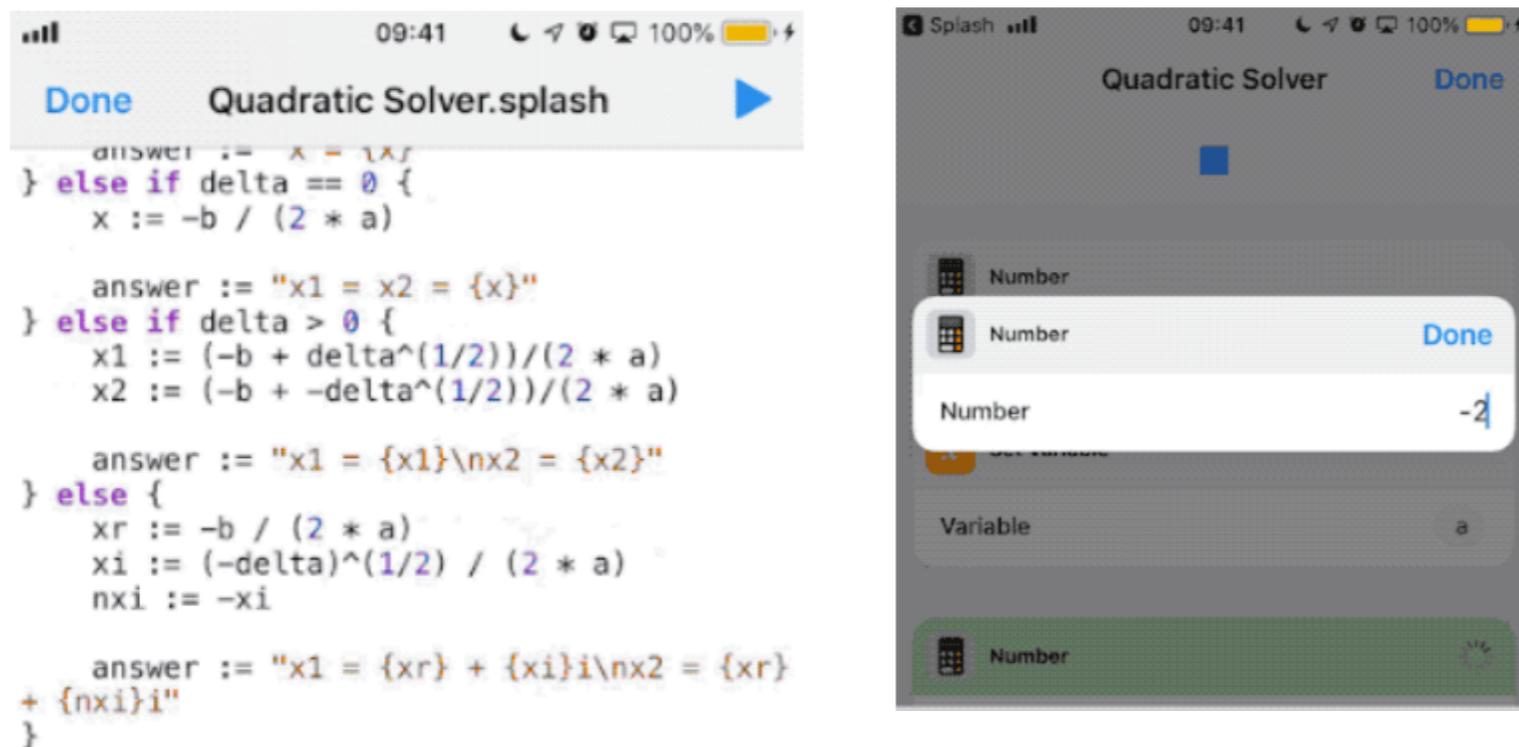
```
[0-9]+.[0-9]* { yylval.value = atof(yytext); return FLOAT; }
[0-9]+ { yylval.value = atoi(yytext); return INTEGER; }
```

# Demo

<https://github.com/jraufeisen/Language-Parsing-in-Action>

# What else can we do?

SPLASH : Simple Programming LAnguage for SHortcuts



The image shows two screenshots of the SPLASH app interface. The left screenshot displays the code for a quadratic solver. The right screenshot shows the execution environment where users can input values for variables like 'a' and 'b' to solve the equation.

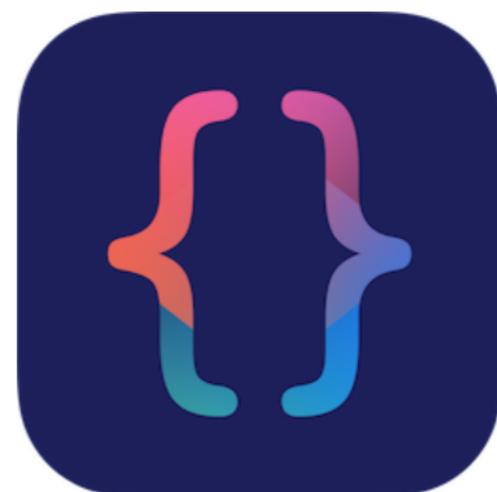
```
Done Quadratic Solver.splash 09:41 100% ↻

answer := x - x̄
} else if delta == 0 {
    x := -b / (2 * a)

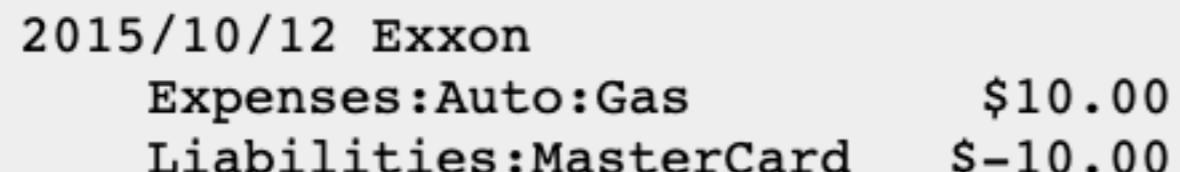
    answer := "x1 = x2 = {x}"
} else if delta > 0 {
    x1 := (-b + delta^(1/2))/(2 * a)
    x2 := (-b - delta^(1/2))/(2 * a)

    answer := "x1 = {x1}\nx2 = {x2}"
} else {
    xr := -b / (2 * a)
    xi := (-delta^(1/2)) / (2 * a)
    nxi := -xi

    answer := "x1 = {xr} + {xi}i\nx2 = {xr}
+ {nxi}i"
}
```



My current project: Budget!



2015/10/12 Exxon  
Expenses:Auto:Gas \$10.00  
Liabilities:MasterCard -\$10.00

