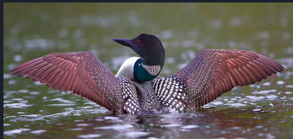
LOON

The Language of Object Notation

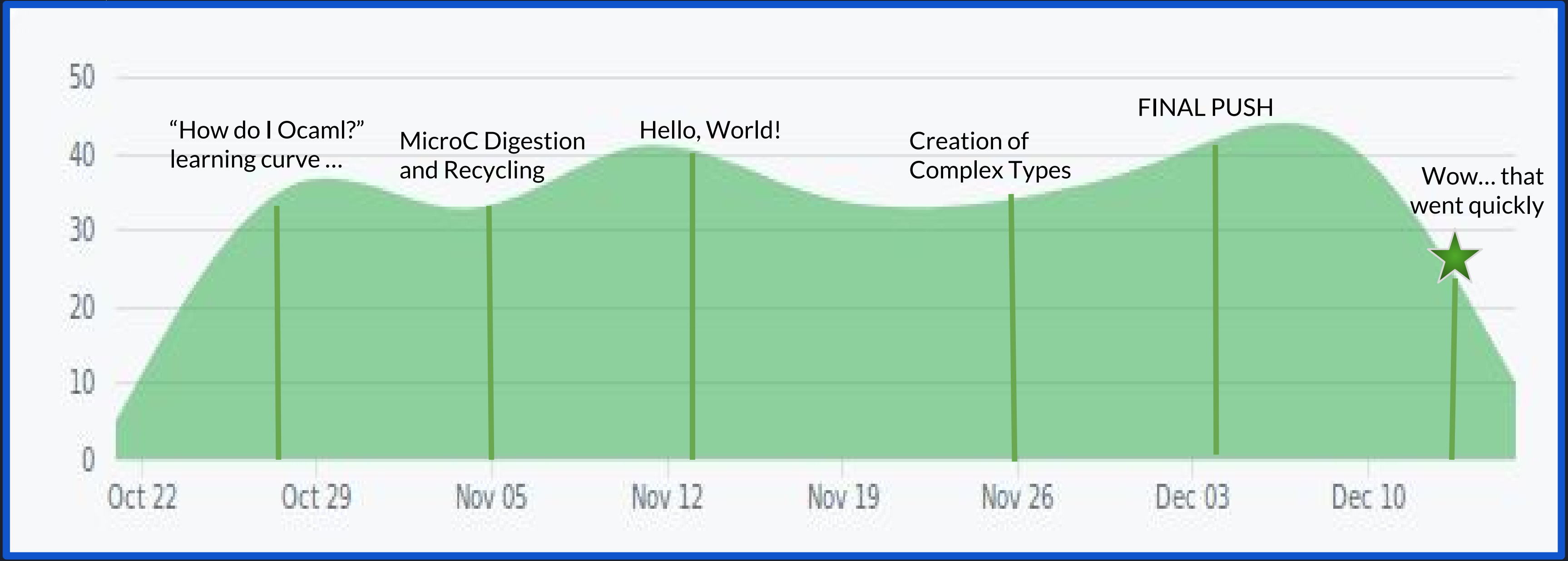
Jack Ricci,
Niles Christensen,
Habin Lee,
Chelci Houston-Burroughs,
Kyle Hughes

Motivation and Language Overview

- Over the past decade, JavaScript Object Notation (JSON) has arguably become the format of choice for transferring data between web applications and service.
- LOON (Language of Object Notation), provides a simple and efficient way to construct and manipulate JSON data for such transfers.



Project Timeline



“How do I Ocaml?” learning curve …

MicroC Digestion and Recycling

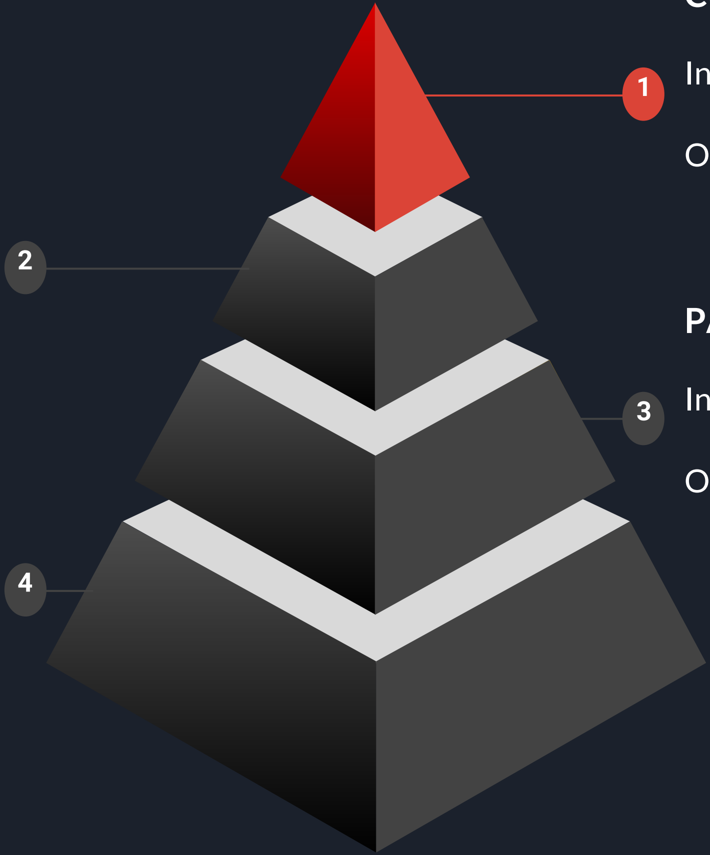
Hello, World!

Creation of Complex Types

FINAL PUSH

Wow… that went quickly

System Architecture



CODEGEN

Input: “SAST”

Output: LLVM IR

SEMANT

Input: AST

Output: “SAST”

PARSER

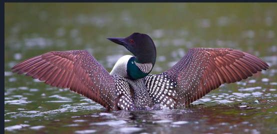
Input: Tokens

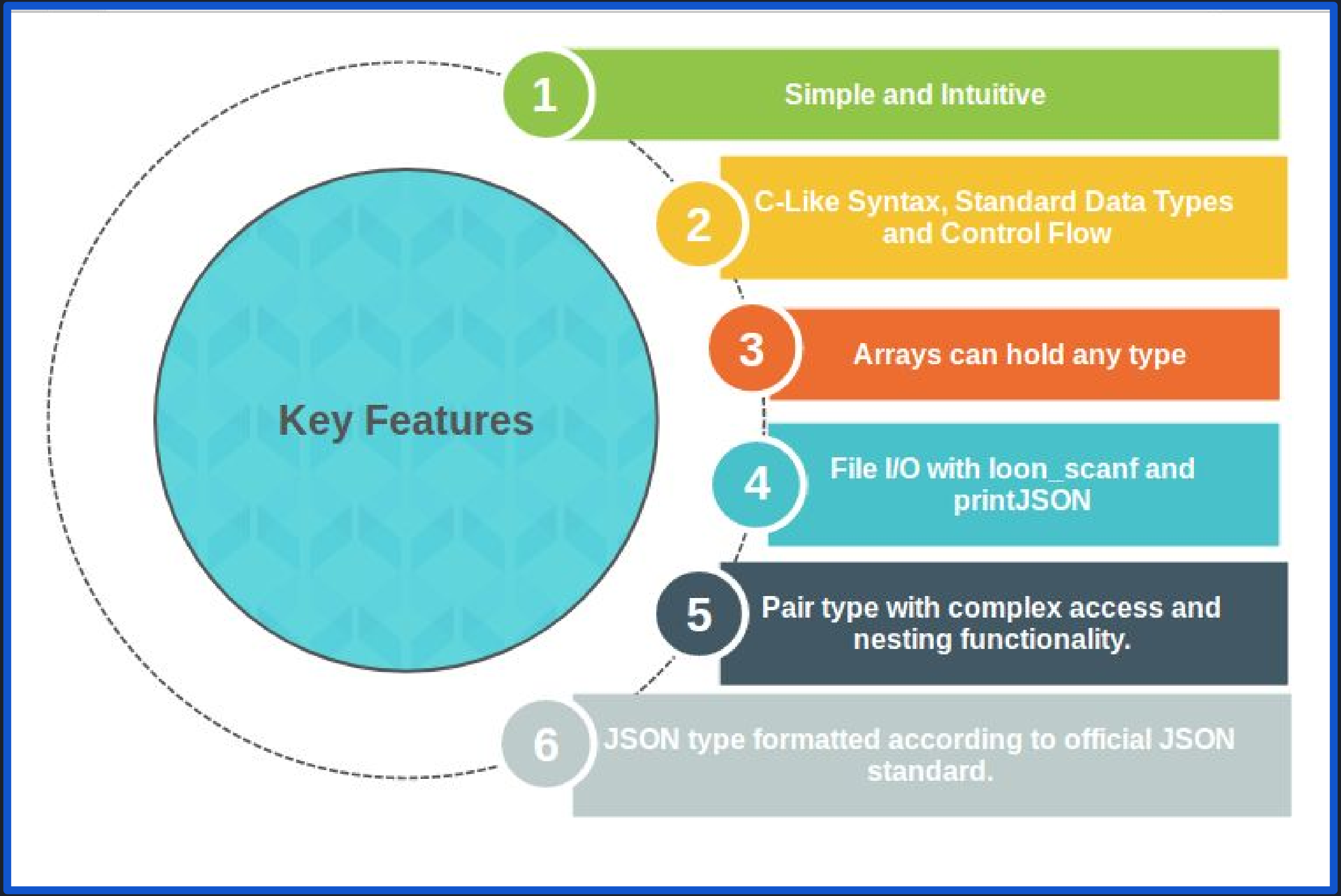
Output: AST

SCANNER

Input: Program Text

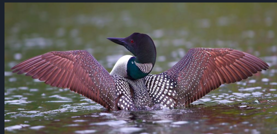
Output: Tokens



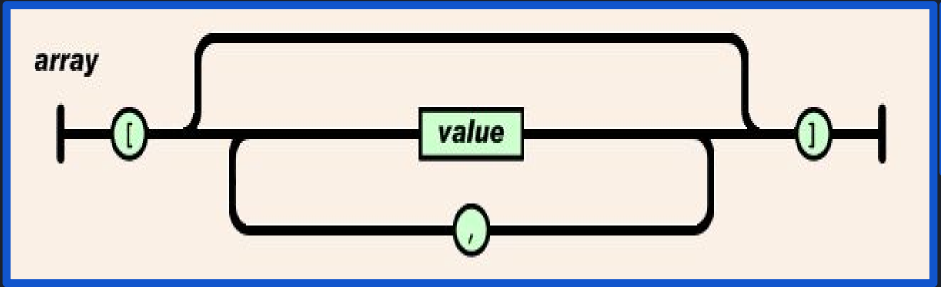


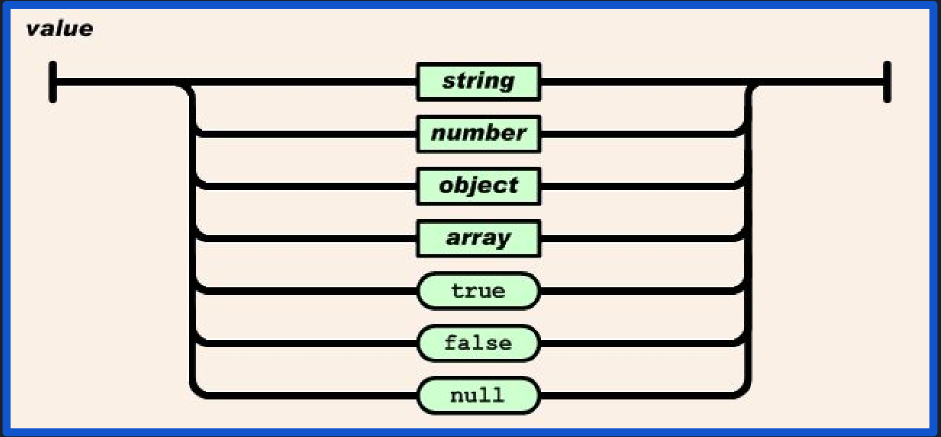
Key Features

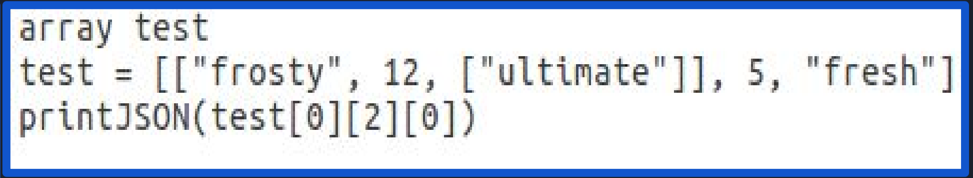
1. Simple and Intuitive
2. C-Like Syntax, Standard Data Types and Control Flow
3. Arrays can hold any type
4. File I/O with loon_scanf and printJSON
5. Pair type with complex access and nesting functionality.
6. JSON type formatted according to official JSON standard.



Array Type







```
array test
test = [["frosty", 12, ["ultimate"]], 5, "fresh"]
printJSON(test[0][2][0])
```

- Can we make the above, syntactically correct JSON list accessible?

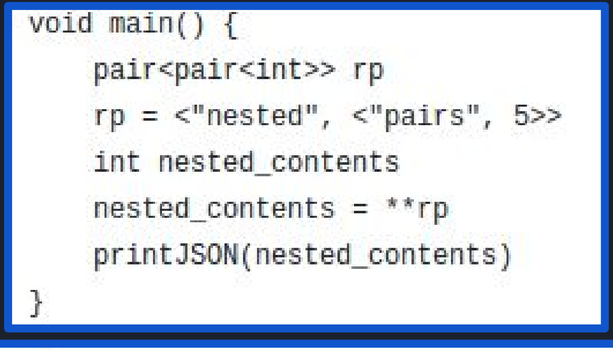
JSON and Pair Types

JSON Advanced Features

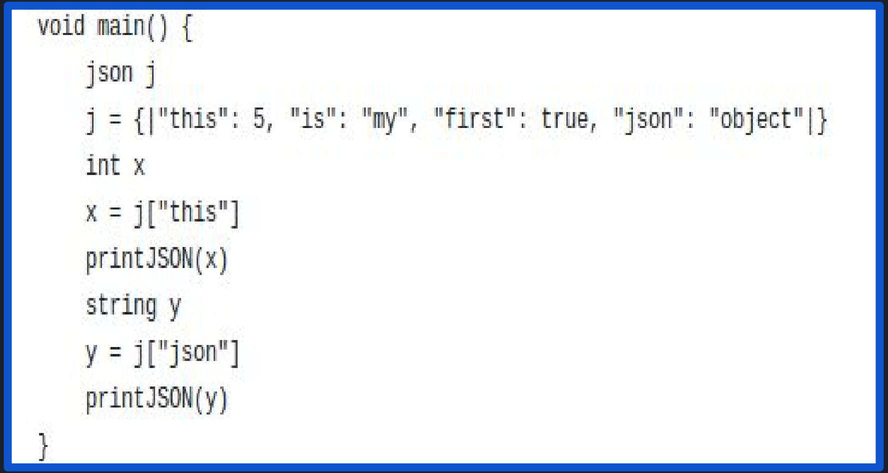
- Can be added to
- Constant time lookup
- Polymorphic value types

Pair Advanced Features

- Storage for a key and a value
- Value can be accessed in code
- Can be added to JSON objects or to a pair to create new JSON



```
void main() {
    pair<pair<int>> rp
    rp = <"nested", <"pairs", 5>>
    int nested_contents
    nested_contents = **rp
    printJSON(nested_contents)
}
```



```
void main() {
    json j
    j = {|"this": 5, "is": "my", "first": true, "json": "object"|}
    int x
    x = j["this"]
    printJSON(x)
    string y
    y = j["json"]
    printJSON(y)
}
```

Test Suite

- *Makefile* links targets and builds.
- *Tests* directory contains all tests & test scripts for compiler parser, scanner, and fail tests.
- New tests incorporated for each new feature
- Test Automation Program using *testall.sh*
- Comprehensively includes success and fail tests

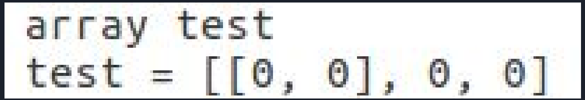
Team Contributions and Lessons Learned

Name	Contributions	Lessons Learned
Jack Ricci	Scanner, Parser, AST, Test Suite, Codegen, Loon_Scanf, printJSON	
Kyle Hughes	Semant, AST, Parser, Test Suite	Software Project Lifecycle, Functional Programming
Chelci Erin Houston-Burroughs	Scanner, Parser, AST, Test Suite, Codegen, Loon_Scanf	Time Management, Clarity in Objectives
Niles Christensen	Scanner, Parser, AST, Test Suite, Codegen, Pair and JSON types	Move fast, break things, test often, and trust in git
Habin Lee	Makefile, AST, Parser, Control	Learn the pace and stick to it, Why we do *tests*

Demos

Arrays demo:

1. Initialize an array containing array and integer objects
2. Transform it into an array containing array and string objects



```
array test
test = [[0, 0], 0, 0]
```

Thank You

Special shout out to our TA Lizzie for her support!