JO

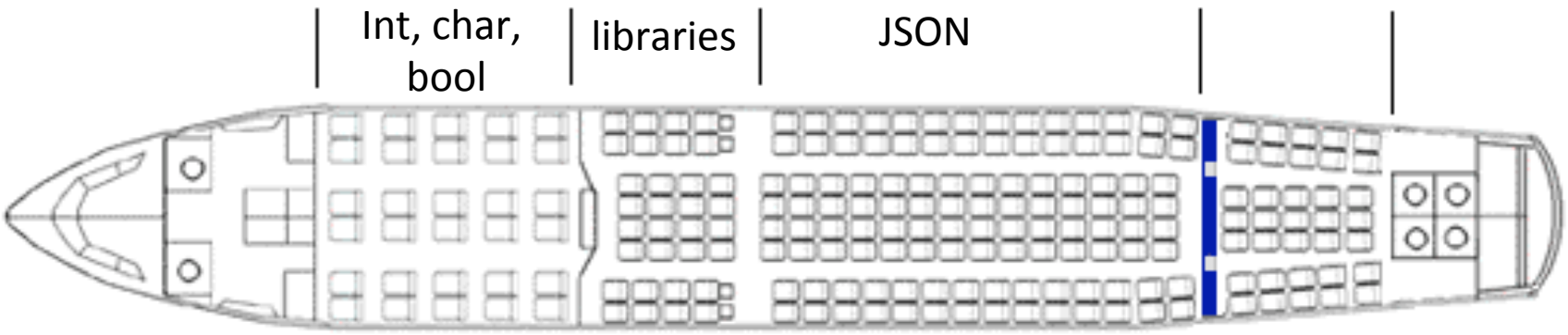
[{ "Name" : "Abhinav Bajaj", "UNI" : "ab3900", "Role" : "System Architect" },

{ "Name" : "Arpit Gupta", "UNI" : "ag3418", "Role" : "Language Guru" },

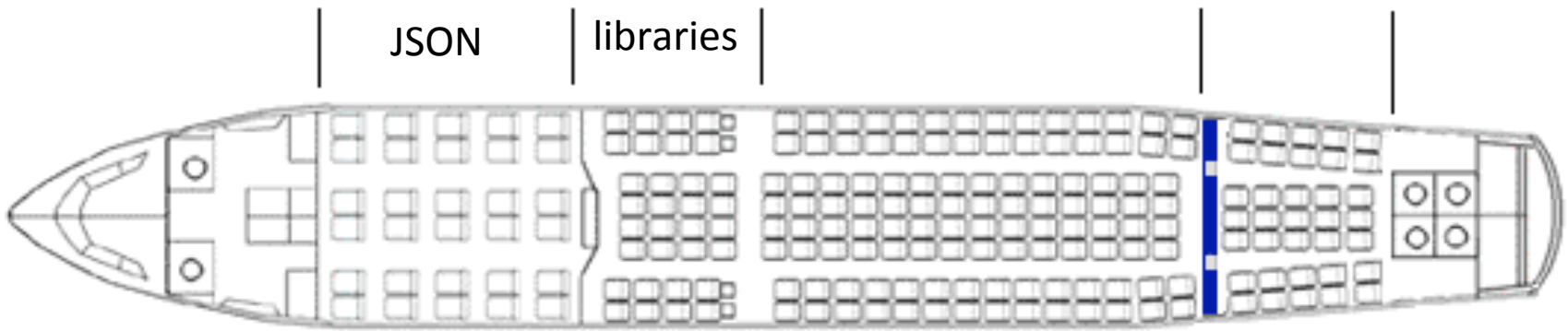
{ "Name" : "Chase Larson", "UNI" : "col2107", "Role" : "Manager" },

{ "Name" : "Sriharsha Gundappa", "UNI" : "sg3163", "Role" : "Verification & Validation" }]

Overview and Motivation



Other Languages' , Seating Arrangement



JO's
Seating Arrangement

Operations on Json and Lists

- json1["Name"] = "Arpit" , stores value "Arpit" for attribute "Name”
- a = json1["Name"] , returns the value at the attribute.
- <json1> - <string>= <json> : removes the key denoted by<string>
- print(<json>) : prettyPrints Json
- <json1> == <json2> : returns true if the json match perfectly
- <list1> ++ <list2> = <list1+list2>
- <object1> + <object2> returns a list : [<object1> , <object2>]

In-Built functions

- typeStruct() : Structure of json
  - a = #{"info" : {"name" : "arpit" , "subjects" : ["plt","algo"]}}#
  - a.typeStruct() returns the string :
    - {string : { string: string, string : list } }
- attrList() : returns an attribute list of Json
- makeString()
- read() : directly reads a json

Too many API calls!

```
func returnOneMillionCalls()
      json1 = read("json1.txt")
      json2 = read("json2.txt")
      json3 = read("json3.txt")
      json4 = read("json4.txt")
      jsonList = json1 + json2 + json3 + json4
      jsonFinal = #{}#
      jsonFinal[" all"] = jsonList
      return jsonFinal
  end
```

Tutorial

- Program Execution – Starts in main()

```
func main ()
        [your code]
end
```

- Variable Declaration – Type Inferred

Type	Declaration
String	a = "Hello World"
Number	b = 10
JSON	c = #{"name":"harsha"}#
List	d = [4,6,"seven"]
Boolean	e = true
null	f = null

• Operators

Comparison operators work on any type. For e.g if a = 3 and b = 2

Equality	**a == b**	**returns false**
Not Equals	a != b	returns true

- Operation on Numbers

Addition	**a ++ b**	**returns 5**
Subtraction	a -- b	returns 1
Multiplication	a ** b	returns 6
Division	a // b	returns 1.5
Modulo	a %% b	returns 1
Greater Than	a > b	returns true
Less Than	a < b	returns false
Greater Than Equal To	a >= b	return true
Less Than Equal To	a <= b	return false

- Concatenation

Concatenation Type	Example	Result
String Concatenation	If a = "Hello " and b = "World" a ++ b	"Hello World"
List Concatenation	If a = ["a", "b", "c"] and b = ["d"] a ++ b	["a", "b", "c", "d"]
List Concatenation with other types	If a = "a" and b = ["b", "c”] a + b	["a", ["b", "c"]]

- Print – prints any type – print(obj)

If a = 5 and b = #{"name":"harsha","courses":[1,"PLT",true]}#

Print Type	Usage	Output
Print Number, String, Bool	print(a)	5

Print JSON (Pretty Prints)	print(b)	{
		"courses":[
		1,
		"PLT",
		true
		],
		"name":"harsha"
		}

Print List	print([1,2,3])	[1,2,3]

- Operations on Non-Primitive Types and its function

Type	Example	Result
Json Element Access	If a = #{"name":"harsha","courses":[1,"PLT",true]}# a["name"]	"harsha"
List Element Access	If a = [1,2,3,4] a[1]	2
Json Element Assignment	If a = #{"a":"b"}# a["c"] = "d"	{"a":"b", "c":"d”}
List Element Assignment	If a = [1,2,3] a[1] = 3	[1,3,3]
Json Element Removal	#{"a":"b", "c":"d"}# a - "c"	{"a":"b”}
List Element Removal	If a = ["a", "b", "c"] , b = ["b", "c"] , d = "c" a - d	["a", "c"]
List Element Removal from List	If a = ["a", "b", "c"] , b = ["b", "c"] , d = "c" a - b	["a"]
typeStruct() Method	If a = #{"name":"harsha", "number":12223, "list":[], "json": {"name":"harsha"}, "boolean":true}# a.typeStruct()	

- # Control Flow

If… else - if..else.. Statements closes with the keyword " end".
Supports In and Not In

For loop – to iterate over a list, closes with keyword “end”
Supports In and Not In

Function Declaration

Use the "func" keyword to define a function.
The format is: func functionName(arguments).
Close functions with the keyword " end"

GCD Example	func findGCD(a,b)
	  if(b == 0)
	  return a
	  end
	  return findGCD(b, a%%b)
	end

e.g. if (5 == 5)
  print ("true")
else
  print ("false")
End
	true

l = [1,2,3,4]
if (5 in l)
  print ("true")
else
  print ("false")
end
	false

If a = [2,3,4] for i in a print (i) end	234

Evolution of the language (Stage 1)

```
decl a = "Hello World"
func main ()
print(a)
return mainfunc
end
```

- All variable declaration started with "decl"
- Print in-built function was implemented

Evolution of the language (Stage 2)

```
decl x = 5;
decl y = "abhinav";
decl a = #{"class":["PLT","OS"]}#;
decl b = [34,45,56];
decl c = true;
```

- Added support for json and list and bool and Number types
- Json and List were treated as string
- All parsing of Json and List left to backend C++
- Used an open source library to parse and create json object

Evolution of the language (Stage 3)

```
decl a= 45
decl b= 81

func findGCD( a ,b)
if( b == 0)
 return a
end
return findGCD(b, a%%b)
end

func main ()
print( findGCD(a,b) )
end
```

- GCD works. Awesome!!!
- Almost all operators working
- weird "return mainfunc" removed
- Parser now parses Json and List
- C++ backend reduced use of library to only parse the json string

Evolution of the language (Stage 4)

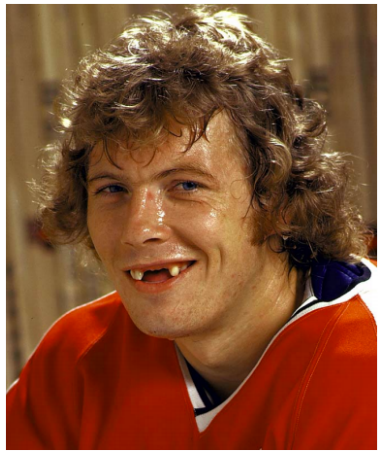
```
func test(l)
     print (l - “d”)
     return null
 end

func main ()
      e = [“a”,”c”,’d”]
      print(type(e))  /* List */
      test(e)
      e = 5
      print(type(e))  /* Number */
end
```

- Dynamic Typing by updating the Symbol table
- Removed "decl" used in declaration
- assignment operation becomes variable declaration

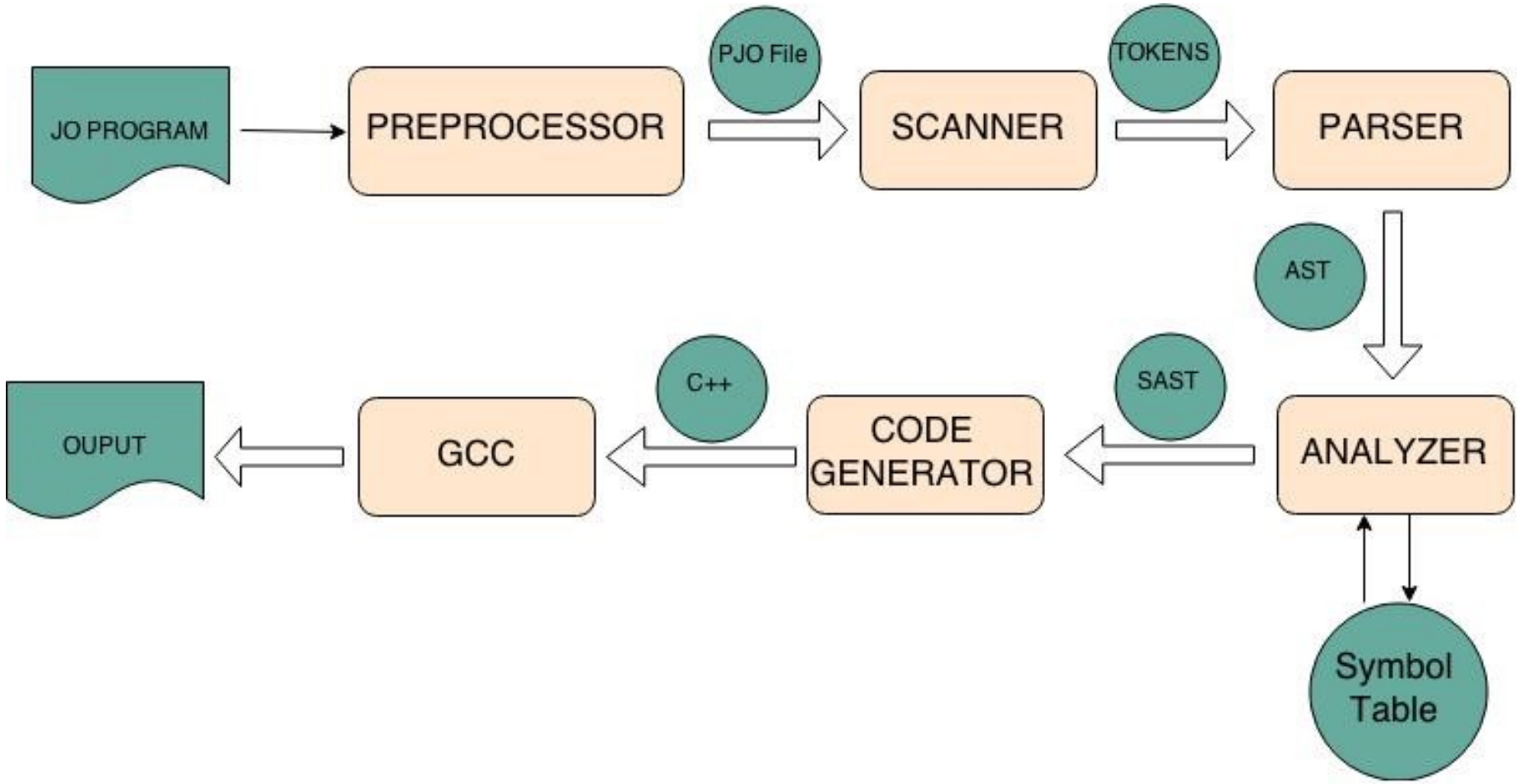
Evolution of the language (Stage 5)

```
func main()

  input = read("testfiles/yelpPlaces.json")
  datalist = input["data"]

  c      = #{"business_id": "Iu-
oeVzv8ZgP18NIB0UMqg", "full_address":   "3320 S
Hill StSouth East LALos Angeles, CA 90007",
"schools":   ["University of Southern California"],
"open": true ,   "categories": ["Medical Centers" ,
" Health and Medical"],    "photo_url": "http: ://
s3-media1.ak.yelpcdn.com/bphoto/
SdUWxREuWuPvvot6faxfXg/ms.jpg", "city": "Los
Angeles", "review_count": 2, "name": "Southern
California Medical Group", "neighborhoods": ["South
East LA"], "url": "http: ://www.yelp.com/biz/
southern-california-medical-group-los-angeles",
"longitude": -118.274281, "state": "CA", "stars":
3.5, "latitude": 34.0197100000000003, "type":
"business"}#

  if ((c in datalist)
       print("found")
  end
end
```

- Realized we dont have support for negative or decimal number (Last night!!!)
- Ocaml string_of_float gives 5.0 => "5." and killed us



Architecture Design



Key Lessons

Judge a language by its size, you must not .(# of features)

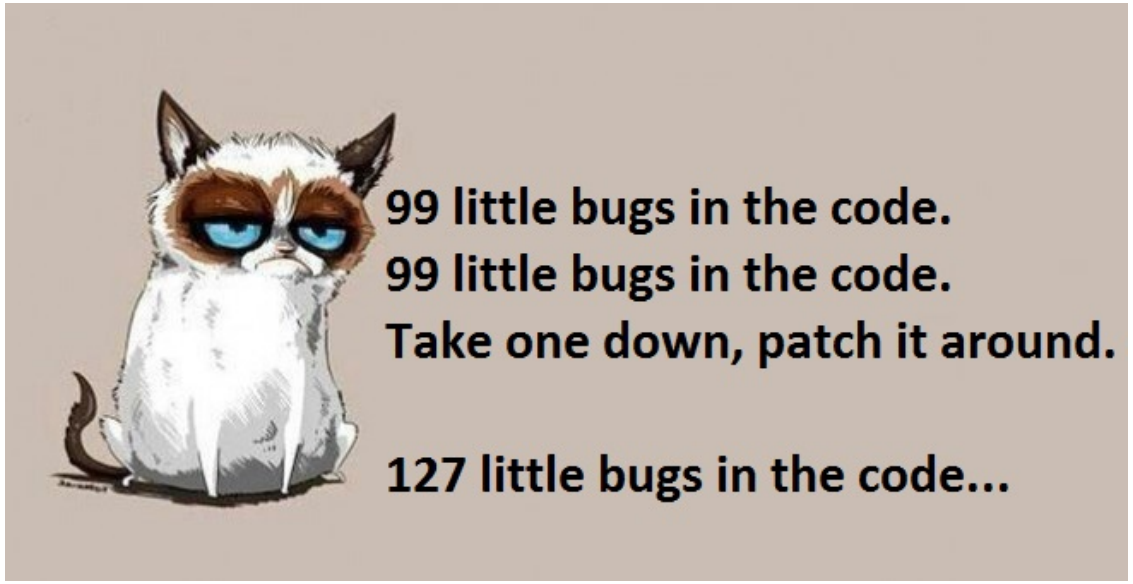
Git rocks

Key Lessons

Let use OCaml

Time/Team Management

Summary


99 little bugs in the code.
99 little bugs in the code.
Take one down, patch it around.
127 little bugs in the code...