QL

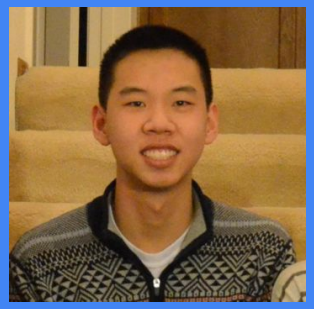
The QL-est language around.

The Coders

Matt Piccolella
Manager

Anshul Gupta
System Architect

Evan Tarrh
Tester



Gary Lin
Language Guru

Mayank Mahajan
System Integrator

Goal: Make JSON great again

QL makes JSON querying easier.

- Typing requires lots of casting
- Nested queries are difficult
- Readability is a concern
- Iteration is unintuitive

Language Features

JSON data type

```
json id = json("json_file")
```

Where clause

```
where (boolean_condition) as id {
  where_body
} in json_array
```

Function declarations

```
function id (parameters) : return_type {
  function_body
}
```

Global Scoping

```
int global = 1
function increment(int b) : int {
  return b + global
}
```

Hello, World

```
{
 "owner" : "Matt",
 "friends" : [
   {
     "name" : "Anshul",
     "age" : 12
   },
   {
     "name" : "Evan",
     "age" : 54
   },
   {
     "name" : "Gary",
     "age" : 21
   },
   {
     "name" : "Mayank",
     "age" : 32
   }
 ],
}
```

```
json test = json("sample.json")
where (elem["age"] > 20) as elem {
  string s = elem["name"]
  print(s)
} in test["friends"]
```

Inferred Types

Problem: Java requires explicit types, but QL doesn't.
Solution: Infer types from statements and expressions, map JSON selectors to QL types.

```
#~~ 1. Infer from operations. ~~#
json a = json("my_file")
string x = "matt" + "is" + a["num1"]
float y = 1.0 + a["num2"]
string z = a["num1"]

#~~ 2. Infer from function passing. ~~#
function increment(int c) : int {
  int x = c + 10
  return x
}
json a = json("dude.json")
increment(a["dude"])
```

```
#~~ 3. Infer from boolean operations. ~~#
json a = json("hello.json")
if (a["matt"] > 5) {
  print("hello")
}
string f = a["matt"]

#~~ 4. Infer from array and JSON access. ~~#
json a = json("sample.json")
array float y = [4.3; 5.0; 1.2]
float z = y[a["int_index"]]
int xx = a["int_index"]
```

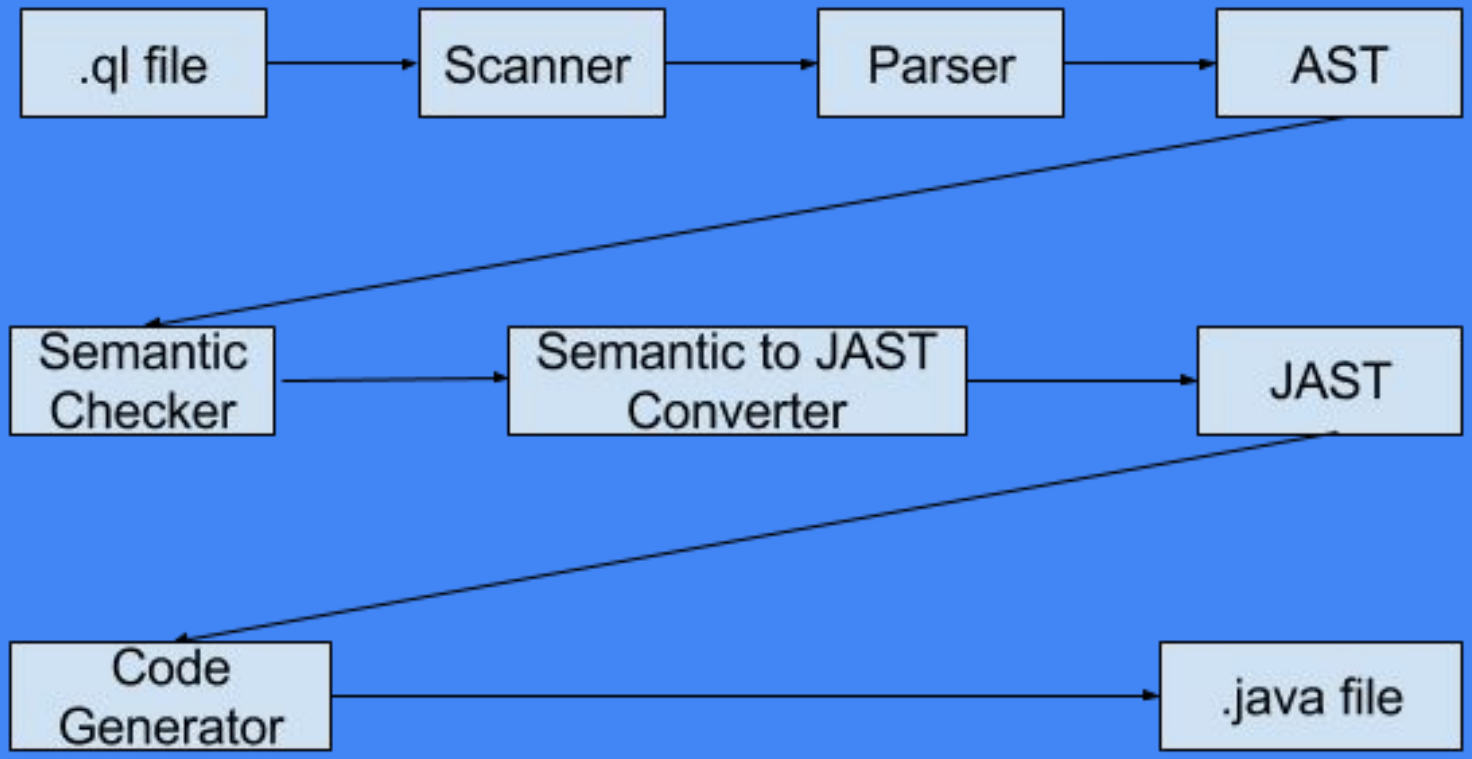
JSON Code Generation

Problem: We need a type at each access for nested queries.
Solution : Iterate through the type of each selector and do in-line casting between JSONObject and JSONArray.

```
json a = json("sample.json")
int b = a["friends"][1]["age"]
string c = a["friends"][2]["name"]
```

```
JSONObject a = (JSONObject) (new JSONParser()).parse(new FileReader("sample.json"));
int b = ((Long) ((JSONObject) ((JSONArray) a.get("friends")).get(1)).get("age")).intValue();
String c = (String)((JSONObject) ((JSONArray) a.get("friends")).get(2)).get("name");
```

System Architecture

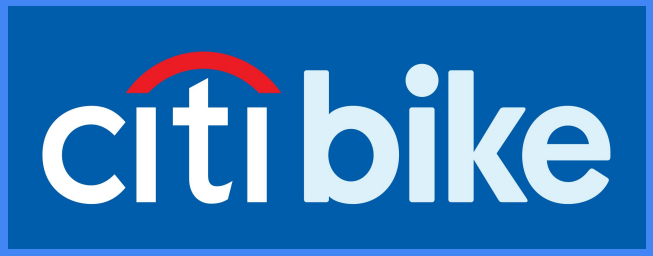


Test Suite

Feature	No. of tests
Arrays	9
Assignment	9
Booleans	4
For Loops	6
Functions	15
“Hello, world!”	2
If/Else	9
JSON	22
Where loops	4
While loops	4
Integration tests	5
Total tests	**90**

Demos


fitbit


citi bike


Columbia University
Directory of Classes