webLang

Ryan Bernstein
Christophe Rimann
Brendan Burke
Jordan Vega
Julian Serra
Inspiration

Web programming sucks in most languages

- Use data types not suited for it.
- Lots of formulaic overhead.
- A lot of unnecessary work for the developer.
What is WebLang?

- Language designed to simplify interactions with RESTful APIs.
  - Aimed at programmers looking to simplify the process of integrating API data into their programs.
  - Handles conventional JSON return types from these APIs.
  - Uses C libraries to interact with servers using HTTP.

- Buzz words:
  - Statically scoped
  - Imperative
  - Semi-statically typed
    - Static when possible, otherwise dynamic.
Software Development Tools

- Code written in Haskell, C, C++, bash, python, and LLVM
  - LLVM via LLVM-hs, LLVM-hs-pure
  - Lexing + Parsing via Alex + Happy
- Communication through Slack
- CI with github, travis
System Architecture
Weblang Design Decisions

- Web-centric: Functions are Endpoints
  - Compiled functions are exposed as a server
  - External endpoints are just used as functions
  - Everything is JSON
  - Functions take and return one JSON argument

- Declarative interface to APIs

- Type system: Some static, some dynamic

- Types as primitive-predicate pairs

```javascript
import {url: "https://hooks.slack.com/services/T74RW7J0N/B891X5YNN/", key: "", secret:"", endpoints: [{fnName:"sendSlackMsg", endpoint:"BaQHlfLTmQNNKHH3EE6PrR1", is_post:true}]

include "slackAPI.wl"

slack arg : Str -> Obj
sendSlackMsg {text: arg}
```
Data Types

- **JSON:** Obj, Arr, Str, Num, Bool, Null
  - Arbitrary nesting of containers

- **Semi-statically typed**
  - Static whenever possible; dynamic whenever not
  - Because we rely on data from the web, we can’t assume types we receive
  - Asserts and pre- and post- conditions

```plaintext
type Int x : Num
   x % 1 == 0

type Pos x : Int
   zero = 0
   x >= zero

type Even s : Pos
   s % 2 == 0

type Odd s : Pos
   (s - 1) % 2 == 0

incOdd x : Odd -> Even
   x + 1

f x : Int -> Even
   y = if x :? Odd
      (incOdd x)
   else
      x
   log y
```
Development Timeline

- Followed class deliverables timeline
- Back and forth with editing components as dependencies and ideas changed
- Weekly “sprints”
- Check ins with TA (Lizzie)

Contributions to master, excluding merge commits
Testing, Continuous Integration, and the Stdlib

- Compared sample programs to expected output text files using a python script.
- Ran the test suite with Travis CI
- Stdlib functions:
  - GCD
  - Bubblesort
  - Contains
  - Average
  - Create Fixed Array
Demo Programs

1. Sending a slack message
2. Activating Travis Build
3. Email -> Text GCD
4. The GRAND finale