



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.

```
#include <stdio.h> /* printf, sprintf */
#include <stdlib.h> /* exit */
#include <unistd.h> /* read, write, close */
#include <string.h> /* memcpy, memset */
#include <sys/socket.h> /* socket, connect */
#include <netinet/in.h> /* struct sockaddr_in, struct sockaddr */
#include <netdb.h> /* struct hostent, gethostbyname */

void error(const char *msg) { perror(msg); exit(0); }

int main(int argc, char *argv[])
{
    /* first what are we going to send and where are we going to send it? */
    int portno = 80;
    char *host = "https://hooks.slack.com/services/T74RW7J0N/B891X5YNN/";
    char *message_fmt = "POST /BaQHifLLTmQQMKH3EE6PrR1 HTTP/1.0\r\n\r\n";

    struct hostent *server;
    struct sockaddr_in serv_addr;
    int sockfd, bytes, sent, received, total;
    char message[1024], response[4096];

    if (argc < 3) { puts("Parameters: <apikey> <command>"); exit(0); }

    /* fill in the parameters */
    sprintf(message, message_fmt, argv[1], argv[2]);
    printf("Request:\r\n%s\r\n", message);

    /* create the socket */
    sockfd = socket(AF_INET, SOCK_STREAM, 0);
    if (sockfd < 0) error("ERROR opening socket");

    /* lookup the ip address */
    server = gethostbyname(host);
    if (server == NULL) error("ERROR, no such host");

    /* fill in the structure */
    memset(&serv_addr, 0, sizeof(serv_addr));
    serv_addr.sin_family = AF_INET;
    serv_addr.sin_port = htons(portno);
    memcpy(&serv_addr.sin_addr, server->h_addr, server->h_length);

    /* connect the socket */
    if (connect(sockfd, (struct sockaddr *)&serv_addr, sizeof(serv_addr)) < 0)
        error("ERROR connecting");

    /* send the request */
    total = strlen(message);
    sent = 0;
    do {
        bytes = write(sockfd, message+sent, total-sent);
        if (bytes < 0)
            error("ERROR writing message to socket");
        if (bytes == 0)
            break;
        sent+=bytes;
    } while (sent < total);

    /* receive the response */
    memset(response, 0, sizeof(response));
    total = sizeof(response)-1;
    received = 0;
    do {
        bytes = read(sockfd, response+received, total-received);
        if (bytes < 0)
            error("ERROR reading response from socket");
        if (bytes == 0)
            break;
        received+=bytes;
    } while (received < total);

    if (received == total)
        error("ERROR storing complete response from socket");

    /* close the socket */
    close(sockfd);

    /* process response */
    printf("Response:\r\n%s\r\n", response);

    return 0;
}
```

```
import {url: "https://hooks.slack.com/services/T74RW7J0N/B891X5YNN/", key: "", secret: "", endpoints: [{fnName: "sendSlackMsg", endpoint: "BaQHifLLTmQQMKH3EE6PrR1", is_post:true}]}

slack arg : Str -> Obj
sendSlackMsg arg
```

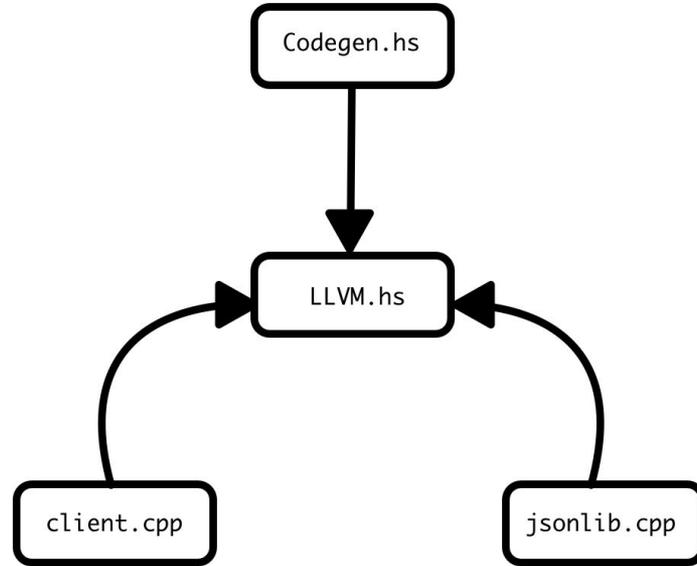
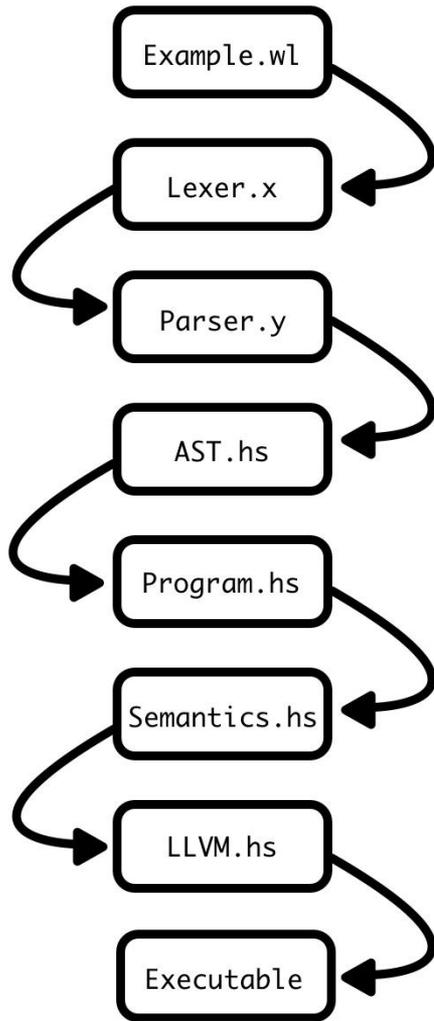
# What is WebLang?

- 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

```
import {url:
  "https://hooks.slack.com/services/T74RW7J0N/B891X5YNN/"
  key: "",
  secret: "",
  endpoints: [{fnName: "sendSlackMsg",
                endpoint: "BaQHlfLLTmQQNKHH3EE6PrR1",
                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

```
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