Jinsen Wu, Matthew Duran, Yanhao Li, Jakob Deutsch, Nicholas Wu

Meet the Team

Matthew Duran
Tester

Yanhao Li
Tester

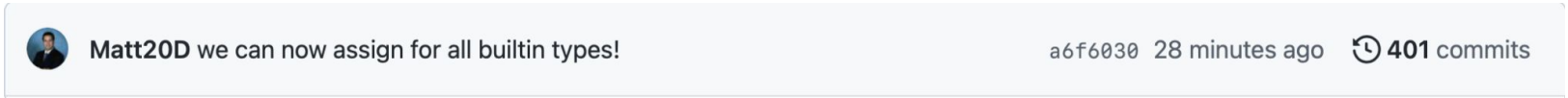
Nicholas Wu
Language Guru

Jakob Deutsch
System Architect

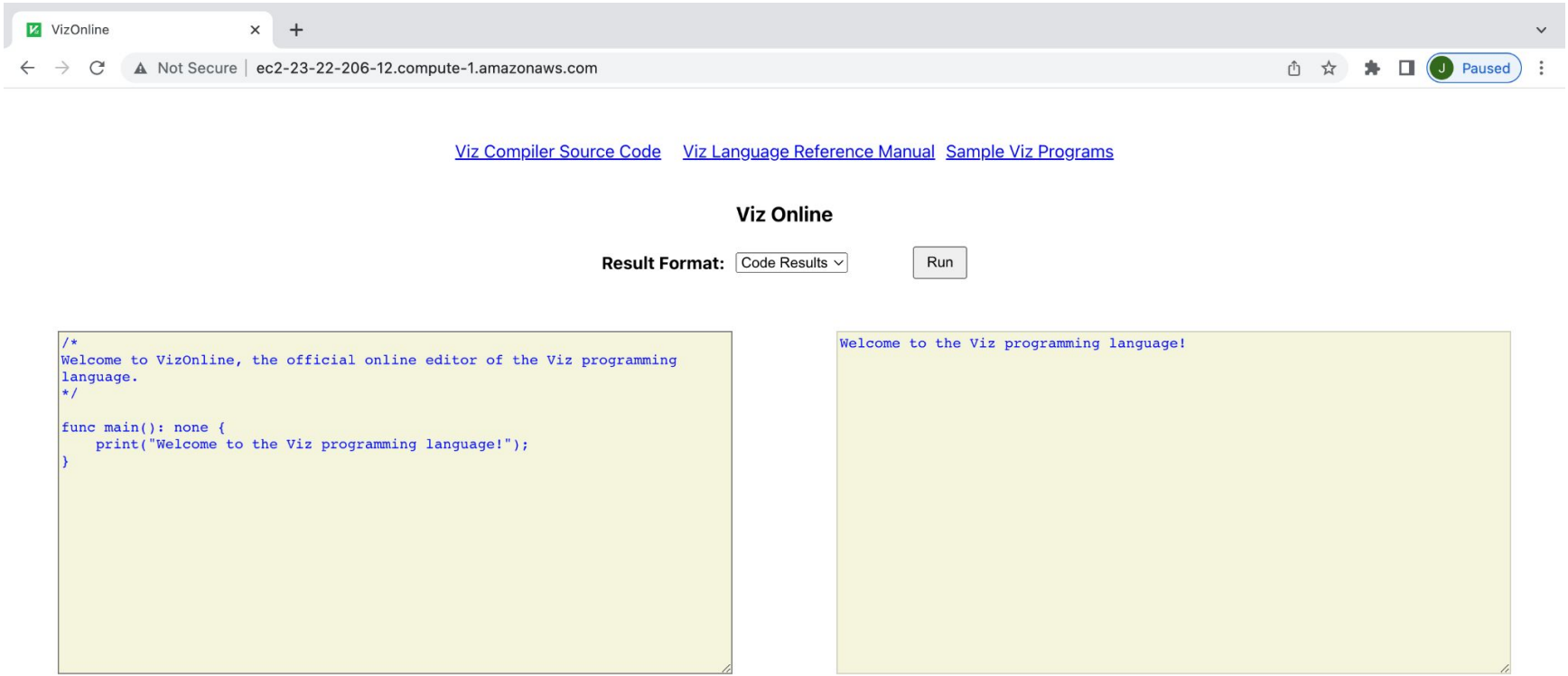
Jinsen Wu
Project Manager

Language Introduction

- General-purpose programing language with visualization of program execution
- Imperative, statically scoped and statically strongly typed
- Simpler features focus on Beginners
- Object oriented structure

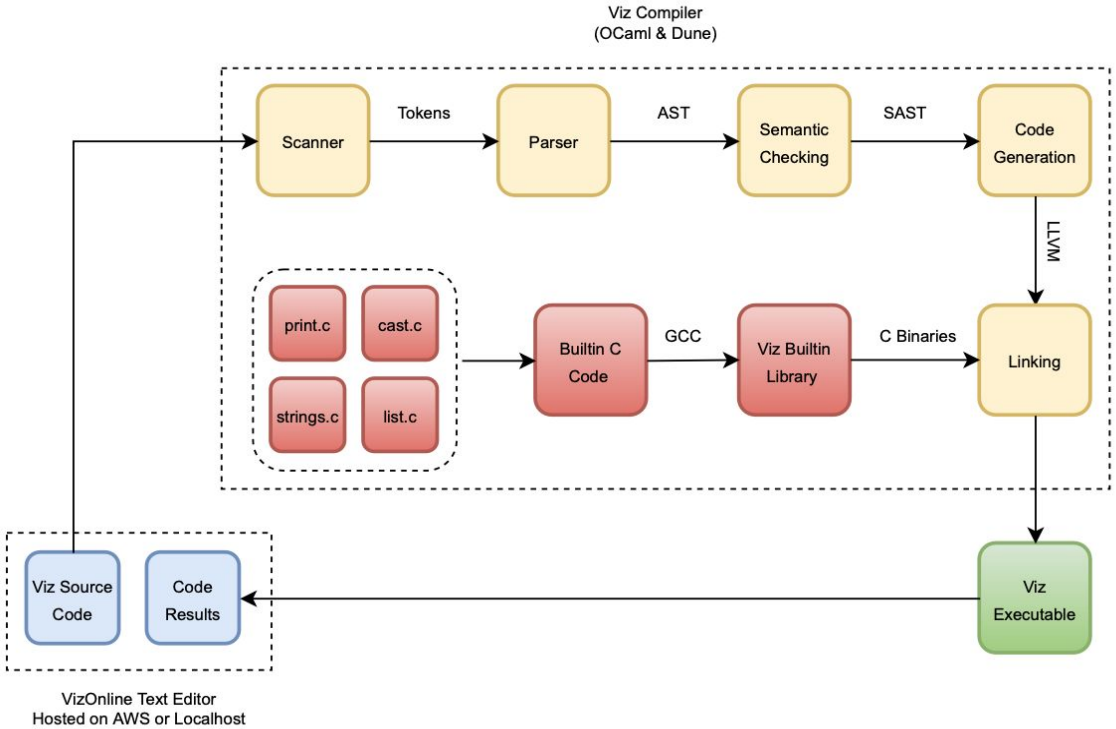


VizOnline Text Editor



http://ec2-23-22-206-12.compute-1.amazonaws.com/

Architectural Design



Basics about Viz

Variable and List

```
@x: bool; // declare
@x = true; // then initialize
@y: string = "good"; // declare and initialize
@y = "bad"; // reassign
@a, @b, @c -> (int, 0); // initializer list where all vars are
ints with value 0
@l: list |int| = [1,2,3,4]; // declare and initialize a list
@l[0] = 1; // reassign a list element
```

Function

```
func hello_world() : none {
  print("Hello World");
}

func main() : none {
   hello_world();
}
```

If/elif/else and loops

```
if (conditional expression) {
       Block...
} elif (conditional expression) {
       Block...
} ... {
       ...
} else {
       Block...
}

// For loop
for k in start...<=end step n {
     Block...
}


// While loop
while boolean_condition {
       Block...
}
```

Coolest Features that we implemented

Object-Oriented Struct

```
struct Person {
    @name: string;
    @age: int;
}

// instantiate a Person object
@my_person: Person;
@my_person.age = 3;
@my_person.name = “Matt”;
```

Typecasting

```
@my_int = |as int| 5.0;
@my_int = |as int| false;
```

Initialize multiple variables with same type and value

```
@x, @y, @z -> (int, 10);
```

Isolated Block Operation

```
func main() : none {
    @x: int = 500;
    print(@x); // 500
    {
        @x: int = 600;
        print(@x); // 600
    };
    print(@x); // 500
}
```

Builtin_functions

```
func main() : none {
    @a: list|int| = [1,1,1,8,8,423,432,432,432,42,32];
    print(list_len(@a)); // 11
    print(pop(@a)); // 32
    print(list_len(@a)); // 10
}
```

C Builtins

Main hurdle to overcome was how to produce functions callable by Viz users but developed in C

Goal: .viz → parsed as func → semantics (I.D. as built in) → LLVM (set up the C call) → C (define the function, the easy part!)

This wasn’t trivial, but we ended up with some big wins! (see next slide)

Builtin Standard Library

- **cast.c**
  - double_to_int, bool_to_int, str_to_int
  - int_to_double, str_to_double
  - bool_to_str, int_to_str, double_to_str
- **print.c**
  - print()
    - print
    - print_int
    - print_float
    - Print_bool
- **strings.c**
  - +
  - str_len
  - to_upper
  - to_lower
  - ==, !=, <, >, <=, >=
- **list.c**
  - list_len_int
  - pop
  - push
  - print_list

Automated Testing Suite

Goal: Create easy to execute and easy to read tests.

Process:

- Step 1: Compose tests that evaluate the correctness of each individual component of our architecture.
- Step 2: Compose more advanced test that test the interaction between different components within the overall application.

Final Product: A simple executable script that runs our 65 practical test programs and produces easy-to-read test results. All you need to do is run ./run_viz_test.sh!

Conclusions and Lessons Learned

- Newfound appreciation for the features that come included in major programming languages like Python, Java, and C/C++.
- A well defined project structure/architecture is key to success.
- Building a language of your own, while difficult, is very rewarding.
- “It is hard to get it to compile, but once it compiles, it works.”

Demo