The Team:
Morris Hopkins - Project Manager
Seungwoo Lee - Language Guru
Lev Brie - System Architect
Alexandros Sigaras - System Integrator
Michal Wolski - Verification and Validation
WHAT IS SWIM?

★ An object-oriented DSL for collecting data from the web
★ Enables access, collection and analysis of web documents out of the box.
★ Easy to learn for users with programming experience
PROPERTIES

Web-centric + Duck Typed

```javascript
url = "example.com";
print(url<"css selector");
```
WHY SWIM?

★ Data scraping has been done before

★ But...They’ve always used specialized libraries...

★ We wanted a DSL for data scraping that worked out of the box

VS.
WHY SWIM?

★ SWIM is for researchers
★ SWIM is for industry
★ SWIM is for general-purpose data analysis

Business  Commercial  Research  IT
SWIM HAS...

★ An Expressive programming syntax

ars = "arstechnica.com";

for each post in ars".post" do
  print(post);
end

print(ars".post");
SWIM HAS...

★ Tremendous Power Under The Hood

for users with more specialized demands, SWIM provides an extensible class and function structure that allows for code reuse, makes use of several aspects of the functional programming paradigm - lambda functions
SYNTACTIC CONSTRUCTS

• Classes

```ruby
class myClass do
  ...
end
```

• Functions

```ruby
fun myFun(params...) do
  ...
end
```
SYNTACTIC CONSTRUCTS

• Built-in Data Types
  • boolean
  • number

• Derived Data Types
  • Lists
  • Dictionaries
  • URLs
  • Strings

• Conditionals
  • if ... elif ... else
Unit Tests

Regression Tests

test_runner.py
Software Development Environment:
Lexing and Parsing: PLY/Python
Version Control: Git with Github hosting
Text Editor: Sublime Text 2
REPL: Custom SWIM REPL
Live Online Editor: swimco.de
Test Suite: Python runner for test suite of SWIM files with integrated regression testing
PROJECT MANAGEMENT

★ Mentoring Sessions with Prof. Aho

★ Weekly Scheduled Meetings + Additional Meetings

★ Google Drive for Documentation Management

★ Google+, Skype, Google Hangouts, Email, Text, etc. for coordinating meetings/remote work
PROJECT MANAGEMENT

Github Commits By Date

# Commits

<table>
<thead>
<tr>
<th>Date</th>
<th># Commits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar 03</td>
<td>0</td>
</tr>
<tr>
<td>Mar 10</td>
<td>0</td>
</tr>
<tr>
<td>Mar 17</td>
<td>0</td>
</tr>
<tr>
<td>Mar 24</td>
<td>0</td>
</tr>
<tr>
<td>Mar 31</td>
<td>0</td>
</tr>
<tr>
<td>Apr 07</td>
<td>0</td>
</tr>
<tr>
<td>Apr 14</td>
<td>0</td>
</tr>
<tr>
<td>Apr 21</td>
<td>0</td>
</tr>
<tr>
<td>Apr 28</td>
<td>0</td>
</tr>
</tbody>
</table>

Wednesday, May 1, 13
PROJECT MANAGEMENT

# Day of the Week

<table>
<thead>
<tr>
<th>Day of the Week</th>
<th>Time of Day</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunday</td>
<td>12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p</td>
</tr>
<tr>
<td>Monday</td>
<td>12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p</td>
</tr>
<tr>
<td>Tuesday</td>
<td>12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p</td>
</tr>
<tr>
<td>Wednesday</td>
<td>12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p</td>
</tr>
<tr>
<td>Thursday</td>
<td>12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p</td>
</tr>
<tr>
<td>Friday</td>
<td>12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p</td>
</tr>
<tr>
<td>Saturday</td>
<td>12a 1a 2a 3a 4a 5a 6a 7a 8a 9a 10a 11a 12p 1p 2p 3p 4p 5p 6p 7p 8p 9p 10p 11p</td>
</tr>
</tbody>
</table>
CONCLUSION

What we would do differently
✴ Plan for scoping right at the beginning

What worked well
✴ AST
✴ Version Control
✴ SWIM REPL

What we learned
✴ Writing your own programming language can be tricky.
✴ The more you take care of right at the beginning, the better!
✴ Create a regression test for everything! Problems will reappear over and over again.
✴ **MAKE IT MODULAR!!! KEEP IT DRY!!**
CONCLUSION

Let’s SWIM!