How not to Design a Scripting Language

Paul Biggar

Department of Computer Science and Statistics
Trinity College Dublin

StackOverflow London, 28th October, 2009

About me

- PhD candidate, Trinity College Dublin
- **Topic:** Compilers, optimizations, scripting languages.

About me

- PhD candidate, Trinity College Dublin
- **Topic:** Compilers, optimizations, scripting languages.

PhD Dissertation

Design and Implementation of an Ahead-of-time PHP Compiler

phc (http://phpcompiler.org)

How not to design a scripting language

- Compilers
- Scripting Languages

How not to design a scripting language

- Compilers
- Scripting Languages

- *Speed*

What is a scripting language?

- Javascript
- Lua
- Perl
- PHP
- Python
- Ruby

What is a scripting language?

- Javascript
- Lua
- Perl
- PHP
- Python
- Ruby

Common Features:

- Dynamic typing
- Duck typing
- Interpreted by default
- FFI via C API

Language implementation

- **Interpreters:** Easy, portable

Language implementation

- **Interpreters:** Easy, portable
- **Compilers:** Not too hard, sometimes portable, *optimizations*

Language implementation

- **Interpreters:** Easy, portable
- **Compilers:** Not too hard, sometimes portable, *optimizations*

NOT THE DRAGON BOOK

Engineering a Compiler by Cooper/Torczon

Modern Compiler Implementation in *X* by Appel

Language implementation

- **Interpreters:** Easy, portable
- **Compilers:** Not too hard, sometimes portable, *optimizations*
- **Just-in-time compilers:** Very difficult, unportable, *fast interpreter*.

What’s right with scripting languages?

What’s right with scripting languages?

1. Elegant and well designed,

What’s right with scripting languages?

1. Elegant and well designed,
2. High level of abstraction,

What's right with scripting languages?

1. Elegant and well designed,
2. High level of abstraction,
3. Dynamic typing (and duck typing).

What’s wrong with scripting languages?

Symptoms: Speed, Portability

What’s wrong with scripting languages?

Symptoms: Speed, Portability

Problem: Language designed for interpreters

- Run-time source code execution

What’s wrong with scripting languages?

Symptoms: Speed, Portability

Problem: Language designed for **one specific** interpreter

- Run-time source code execution
- Foreign Function Interface

FFI

Foreign Function Interface based on CPython interpreter

- Access to C libraries
- Script C applications using Python scripts
- Rewrite hot code in C

FFI (good) implications

- Libraries not that slow
- Can break out of Python for slow code.

FFI (bad) implications

- Language is allowed to be slow
- Must break out of Python for speed.

FFI (worse) implications

- Legacy issues

FFI (worse) implications

- Legacy issues
- Reimplementations

stackoverflow

FFI solution

Don't expose yourself!

- Importing functions into Python with a Domain Specific Language is good

FFI solution

Don’t expose yourself!

- Importing functions into Python with a Domain Specific Language is good
- Only one way of FFI is better

FFI solution

Don't expose yourself!

- Importing functions into Python with a Domain Specific Language is good
- Only one way of FFI is better
- Declarative is best

FFI solution

Don’t expose yourself!

- Importing functions into Python with a Domain Specific Language is good
- Only one way of FFI is better
- Declarative is best

- Any reimplementation can reuse the same libraries without any modifications
- CPython itself can change without hassle

Dynamic source code generation

- `eval` and dynamic `include`/`import`

Dynamic source code generation

- `eval` and dynamic `include`/`import`
  - meta-programming

```
eval (mysql_read (...)[0]);
```

Dynamic source code generation

- `eval` and dynamic `include`/`import`
  - meta-programming
  - `.rc` files

```
username = "myname"
password = "mypass"
server = "srv.domain.com"
```

Dynamic source code generation

- `eval` and dynamic `include`/`import`
  - meta-programming
  - `.rc` files
  - localization

```
$lang = ....;
include ("localisation/locale.$lang.php");
```

Dynamic source code generation

We don’t even know the full program source!!

So they can’t be compiled (ahead-of-time)

Downsides:

- Must use FFI for speed
- Static analysis
- Cool optimizations can’t happen

So they can’t be compiled (ahead-of-time)

Downsides:

- Must use FFI for speed
- Static analysis
- Cool optimizations can’t happen

```
t = ...;
for (i = 0; i < strlen(t); i++)
{
  s[i] = t[i];
}
```

So they can’t be compiled (ahead-of-time)

Downsides:

- Must use FFI for speed
- Static analysis
- Cool optimizations can’t happen

```
t = ...;
_temp = strlen(t);
for (i = 0; i < _temp; i++)
{
   s[i] = t[i];
}
```

So they can't be compiled (ahead-of-time)

Downsides:

- Must use FFI for speed
- Static analysis
- Cool optimizations can't happen

```
alert ($('li').get(0).nodeName);
```

So they can’t be compiled (ahead-of-time)

Downsides:

- Must use FFI for speed
- Static analysis
- Cool optimizations can’t happen

```
alert ($('li')[0].nodeName);
```

JIT compiled

Tracemonkey

http://hacks.mozilla.org/2009/07/tracemonkey-overview/

JIT compiled

Tracemonkey

http://hacks.mozilla.org/2009/07/tracemonkey-overview/

Type Analysis for Javascript

Simon Holm Jensen, Anders Møller and Peter Thiemann
SAS ’09
http://www.brics.dk/TAJS/

Fix at **language** design time

- No dynamic `include`; no `eval`.
    - Compile-time meta-programming

Fix at **language** design time

- No dynamic `include`; no `eval`.
  - Compile-time meta-programming
  - `.rc` files

Fix at **language** design time

- No dynamic `include`; no `eval`.
  - Compile-time meta-programming
  - `.rc` files
  - localization

Doing it right

- Factor
  - compiled model
  - compile-time meta-programming
  - declarative FFI

Open research problems

- Optimizing *boxing*
- High-level optimizations
- Combining ahead-of-time and JIT compilation

Conclusion

Design the next scripting language right