When & How to Win with Programming Languages

@noelwelsh

noelwelsh.com

underscore

Inner Product

Outline

You should open your ❤️ to **more PLs**

PLs are the **biggest driver** of **productivity**

Understand the forces **against adoption**

Understand how and **when** to **adopt**

The **Lawyer Hypothesis**

VIT EXC ERE AR
AM OLU PER TES

SEEK WISDOM

“The first thing we do, let's kill all the lawyers”

William Shakespeare

Lawyers are **not** lizards wearing human skin

Lawyers are **mostly not** lizards wearing human skin

But they do get

paid a lot

David Higgins started his new role in the City office of US **law firm** Kirkland & Ellis … with … a reported **$10m (£7m) salary**.

http://www.cityam.com/265864/citys-best-paid-lawyers-and-they-taking-home

Top lawyers **are actually lawyers**

Software >> Law

Lawyer >> Dev

The **Quest** for **Productivity**

Humans **don’t scale**

Automation
does scale

Productivity ≠ Writing code

Not writing code is the **only way** to be **productive at scale**

Libraries

Languages

Abstraction

Languages >> Libraries

What is a Language **For?**

Control the **machine**

Primitive types

SIMD

Access the OS

Access the **browser**

Access a **framework**

Notation for **expressing solutions**

First-class
values

Generic types

Automatic resource management

“The machine” can be **narrow**

DSL

“**Little**
Language”

Machine determines if language is **used**

Javascript

Notation **upper-bounds** productivity

Where are Languages?

Big languages

Java, Python, Scala, Go, Haskell, Ruby, etc.

Little languages

CSS, CUDA, configuration, Excel, marketing automation, etc.

Little >> Big

Frameworks are libraries

Programmed in

configuration

Most DSLs **don’t realise** they are DSL

CSS Variables ~10 years **too late**

When are Languages **Adopted?**

Big languages

Access

compelling, new “machine”

Javascript:
access **browser**

Objective-C: access **iOS**

Ruby: access

Rails

Respect **legacy** with **better notation**

Scala vs Java

Kotlin vs Java (Android)

Swift vs Objective-C

Rust vs C

Typescript vs Javascript

Cultural fit

Elixir vs Erlang

Go vs every other compiled language

Story time:

Racket

No compelling new machine

No respect for legacy

Story time:
Scala

Compelling

new machine:

Spark

Respect for legacy

Culturally acceptable for me

Little languages

Same forces

but barrier
much lower

CSS, CUDA, configuration, Excel, marketing automation, etc.

Language can be **competitive advantage**

Marketing **automation**

Airtable

Cloudflare

Little >> Big

Frameworks are libraries …

programmed in

configuration

Most DSLs **don't realise** they are DSL

CSS Variables
~10 years **too late**

How are Languages **Adopted?**

Big languages

Consultancy?

Forget it.

Product? **Go all in.**

High cost of failure

Hire **enthusiast early adopters**

Create a **remote first culture**

Prepare to carry
the **burden …**

of **maintaining libraries**

of **maintaining community presence**

of **creating legacy**

Start **small**

Demonstrate

success

Slowly **spread**

Consider **external mentors**

Little languages

Go for it!

Low cost of failure

But **consider …**

Simple (but maybe **not easy**)

Used **enough**

Reasoning across boundaries

Conclusions

Languages are
a **powerful tool**

Conditions
must be right

Add this to your toolbox

@noelwelsh

noelwelsh.com

underscore

Inner Product