

LIS-5364

More On Languages

# The few things 99% of languages have in common:

- Written in plaintext (there are exceptions)
- Punishingly difficult and picky syntax. One character wrong and the whole thing explodes.
- this is important to remember:

Written BY humans FOR humans. Most everything you see isn't some weird computer artifact: it's a choice.

# Things people debate about over languages:

- Human readability?
  - Compiled/interpreted/ or scripted?
  - “Paradigm?” Procedural/Functional/Object-Oriented?
  
  - What was it originally designed to do?
  - How does it get changed?
  - What does it do now?
- Many of these boil down to: SIMPLE vs EASY

SIMPLE and EASY are OPPOSITES

(that's weird)

## SIMPLE: (Think “command line”)

- Few assumptions
- You can see “everything”
- Usually “text-only”
- Rigid Framework
- “Mathematical”
- Few abstractions – simple data types.
- MORE verbose, MORE language to get things done.

## EASY: (Think “iPhone” or “Siri”)

- Many assumptions
- A lot is hidden to simplify the view
- Not always text-only
- Multiple Pre-Loaded Frameworks
- “Visual”
- Many abstractions – simple data types.
- Less language to get something up and running.

# Human Readability ("Matrix" vs. "English")

Old school - Human readability is not important

- Short abstractions are concise and thus quicker
- Forced whitespace is limiting
- There should be *MANY* ways to do a thing.

# Human Readability ("Matrix" vs. "English")

New school – Hey, looks like human readability is at least a little important:

- Multiple people working on projects
- Older code needs to be understandable
- There is value in forcing people to do things only one way

*"Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live."*



# “Closeness” to machine

**Compiled**– Before running, you have to “convert” it. Usually Makes for faster/more efficient code.

**Interpreted** – “Conversion” happens on the fly.

**Scripted** – No “conversion” necessary. Usually only good for shorter/smaller/ scripted things.

# Compiled Languages

- Older
- “Normal”
- “Close to machine”

# “COMPILED” - Source Code



# Compiled Code (binaries)



# INTERPRETED LANGUAGES:

- Basically, they split the difference; interpreted on the fly. **MOST WEB LANGUAGES** are this:

PHP and Javascript!

# Important “real life” Categories. (these are fuzzy)

- **Procedural (default)** - Do things step by step
- **Functional** - Turn it into straight up math. No variables, no “procedures”
- **Object Oriented** - Everything uses an “object” metaphor. (“Black Box Approach”)

# Types of Languages

- Markup Languages (aren't really languages)
- “Normal” Text oriented languages
- “Web” oriented languages
- Specialized languages

# HTML/CSS

- Not really languages. ( Can't really do anything besides dress up and move around text and pictures and other things)
- Traditionally, for dressing up text because fonts and colors (and [HYPERLINKS](#) WHOA) used to be a pretty big deal.
- Today, not even a great choice if you need a website, BUT you should know it because it's now the **FRAMEWORK** for the web.



# Other “Markups”

Markdown

Org-Mode

Zim (perhaps best program ever written)

LaTeX

“Normal” languages following

# Lisp (and other functionals)

(Haskell, Clojure)

- MATHY
- VERY, perhaps TOO “versatile”
- Stallman and Emacs
- Functional is seeing a resurgence, because of its mathematical “purity.” There are NO VARIABLES, WHOA.

# C (C++, C#)

- Practical granddaddy of (modern) everything
- Still a good choice for speed and control, at the cost of ease
- "Why C sucks" -- TOO MUCH control, too much "hardcore stuff" to worry about  
(e.g. "malloc")  
(*"Go" and "Rust" are looking to take on c*)

# Python

- Arguably – the best “all around” language?
- Forced whitespace
- Usually "one way to do a thing"
- In the "middle" on just about all parameters.

# Ruby

Designed for elegance, ease of use, but still powerful

- Highly OO
- Very easy to get started
- Very easy to read, yet still concise.
- Poignant's guide to ruby
- Arguably not quite as fast as slightly older languages
- "Why Ruby Sucks" – Turns out to not be fast enough, probably getting eaten by Javascript?

# Java

- First attempt at dethroning C, looks a lot like it
- Originally company driven, as a result got popular/useful quickly
- interpreted, not compiled (Virtual Machine required)
- "Why Java Sucks" - slow, verbose, too many options/fragmented

Simple v. Easy v. the third thing:

WE NEED CODE ON **THE WEB** LIKE RIGHT NOW.

NOW!!!!!!

**NOWWWW!!!**

(this explains most of the weirdness with – Flash, PHP, and Javascript)



# SERVER-SIDE

Code resides on server

Code is executed by server

Dynamic content is produced

(Ruby on Rails, Django, PHP\*, Wt)

Note why PHP is a little weird. Instead of independent code, PHP code is EMBEDDED IN HTML, but run by the server -- where you'd usually expect it to be the other way around, i.e., have your code PRODUCE html. This also might be why it's so popular.

# PHP

- Designed to handle the web and HTML
- Features were added as needed, not from ground up
- "Why PHP Sucks" Purists HATE IT; probably the most duct-tapiest language of all  
(still, facebook and wordpress)  
(ASP is Microsoft's slightly different "PHP")

# CLIENT SIDE

## CLIENT SIDE APPROACHES

### Java/Flash

Download their little programs, and let them run in your browser

(yes, this can be as dangerous as it sounds)

### Javascript

CODE is EMBEDDED in the html and RUN IN/BY YOUR BROWSER.

(yes, this is even more dangerous than the above. Also very useful. Get NoScript.) -

(greasemonkey is cool, though)

### Silverlight- M\$'s flash

HTML5 - will hopefully save us all. Reimplementing all the good stuff in flash, openly.

# Javascript

- Though looks like c, VERY DIFFERENT FROM JAVA. Confusing, huh?
- CLIENT SIDE (mostly).
- JAVASCRIPT IS EATING THE WEB RIGHT NOW. LOOKING VERY DOMINANT.

# Specialized Languages

For Kids - Scratch

For special functions - Sonic Pi, MySQL

For LOLS - LOLCODE, Shakespeare

For insanity - Brainf\*\*k, Whitespace