Node.js

The good, the bad, and the ugly.

Node?

A set of async IO libraries shipped alongside Google’s V8.

The Good	The Bad

The Good	The Bad
Fast	

The Good: Fast

- Built-in async I/O
    - Node is clever about
- Can achieve with other libraries/frameworks
    - Like ReactPHP. Give PHP some love <3

The Good	The Bad
Fast	
npm	

The Good: npm

Contrary to the belief of many, "npm" is not in fact an abbreviation for "Node Package Manager". It is a recursive bacronymic abbreviation for "npm is not an acronym". (If it was "ninaa", then it would be an acronym, and thus incorrectly named.)

https://docs.npmjs.com/misc/faq#if-npm-is-an-acronym-why-is-it-never-capitalized-

The Good: npm

- It ships with node.js and has from a very early version (0.6.3) and it's an integral part of using node, so there's no fragmentation and strong network effects.
- Using and publishing is easy.
- The website is simple, clean, and provides helpful stats.
- All packages link to the github repo.
- Learns from earlier PMs.

The Good: npm

local package installation

The Good: npm

LOCAL PACKAGE INSTALLATION

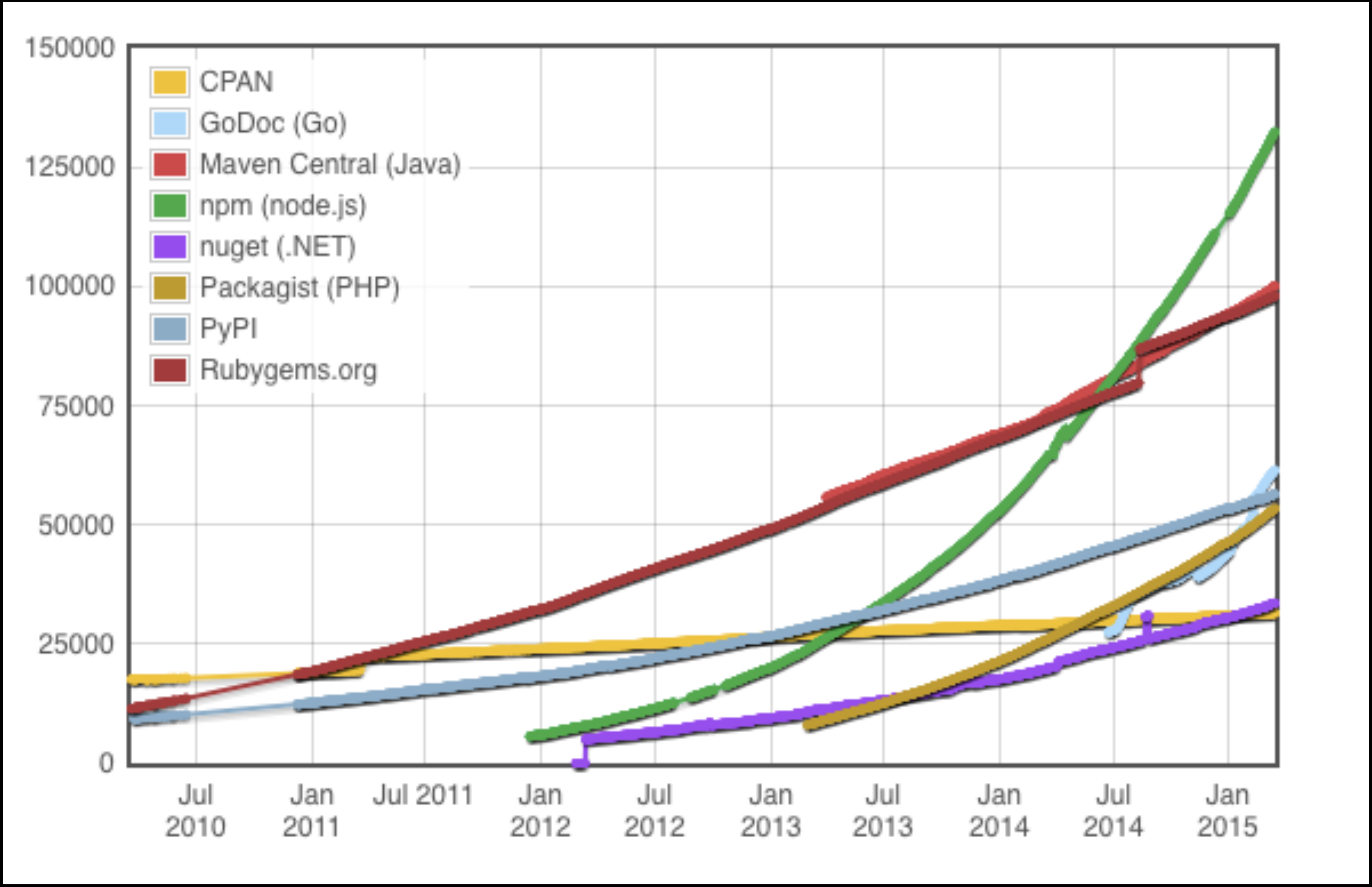
The Good: npm

LOCAL PACKAGE INSTALLATION

The Good: NPM

- No sudo access to install packages.
- Self containment. No dependency conflicts.
- But allows for global installs & bins.

The Good: NPM


150000
125000
100000
75000
50000
25000
0
CPAN
GoDoc (Go)
Maven Central (Java)
npm (node.js)
nuget (.NET)
Packagist (PHP)
PyPI
Rubygems.org
Jul 2010
Jan 2011
Jul 2011
Jan 2012
Jul 2012
Jan 2013
Jul 2013
Jan 2014
Jul 2014
Jan 2015

The Good: NPM

248/day

http://www.modulecounts.com

The Good	The Bad
Fast	
npm	
…	

The Good	The Bad
Fast	
NPM	
……………	

The Good	**The Bad**
Fast	
NPM	
JavaScript	

The Good: JavaScript

- Straight-forward.
- Everywhere.
- Imperative + some functional McJiggs.

The Good: JavaScript

JavaScript IS NOT THE
DOM/BROWSER/INTERNET EXPLORER

The Good	The Bad
Fast	
NPM	
JavaScript	

The Good	The Bad
Fast	
NPM	
JavaScript	JavaScript

The Bad: JavaScript

“JavaScript practices an extremely aggressive type coercion doctrine where comparing apples with bananas always makes sense, especially when they are actually oranges.”

The Bad: JavaScript

“Everything works somehow. Or not, depending on how you look at it and what time of the day it is.”

The Bad: JavaScript

https://www.destroyallsoftware.com/talks/wat

@1:20

The Bad: JavaScript

11.9.3 The Abstract Equality Comparison Algorithm

The comparison x == y, where x and y are values, produces true or false. Such a comparison is performed as follows:

```
1   If Type(x) is the same as Type(y), then
      a   If Type(x) is Undefined, return true.
      b   If Type(x) is Null, return true.
      c   If Type(x) is Number, then
            i    If x is NaN, return false.
            ii   If y is NaN, return false.
            iii  If x is the same Number value as y, return true.
            iv   If x is +0 and y is −0, return true.
            v    If x is −0 and y is +0, return true.
            vi   Return false.
      d   If Type(x) is String, then return true if x and y are exactly the same
          sequence of characters (same length and same characters in
          corresponding positions). Otherwise, return false.
      e   If Type(x) is Boolean, return true if x and y are both true or both
          false. Otherwise, return false.
      f   Return true if x and y refer to the same object. Otherwise, return
          false.
2   If x is null and y is undefined, return true.
3   If x is undefined and y is null, return true.
4   If Type(x) is Number and Type(y) is String,
5   return the result of the comparison x == ToNumber(y).
6   If Type(x) is String and Type(y) is Number,
7   return the result of the comparison ToNumber(x) == y.
8   If Type(x) is Boolean, return the result of the comparison ToNumber(x)
    == y.
9   If Type(y) is Boolean, return the result of the comparison x ==
    ToNumber(y).
10  If Type(x) is either String or Number and Type(y) is Object,
```

The Bad: JavaScript

```
> var xs = ['10', '10', '10']
undefined
> xs
[ '10', '10', '10' ]
> xs.map(parseInt)
[ 10, NaN, 2 ]
>
```

The Bad: JavaScript

"In 1995, Netscape hired Brendan Eich with the promise of letting him implement Scheme (a Lisp dialect) in the browser."

The Bad: JavaScript

Sun

microsystems

The Bad: JavaScript

“We aimed to provide a “glue language” for the Web designers and part time programmers who were building Web content from components such as images, plugins, and Java applets. We saw Java as the “component language” used by higher-priced programmers, where the glue programmers—the Web page designers—would assemble components and automate their interactions using [a scripting language].”

THERE IS HOPE

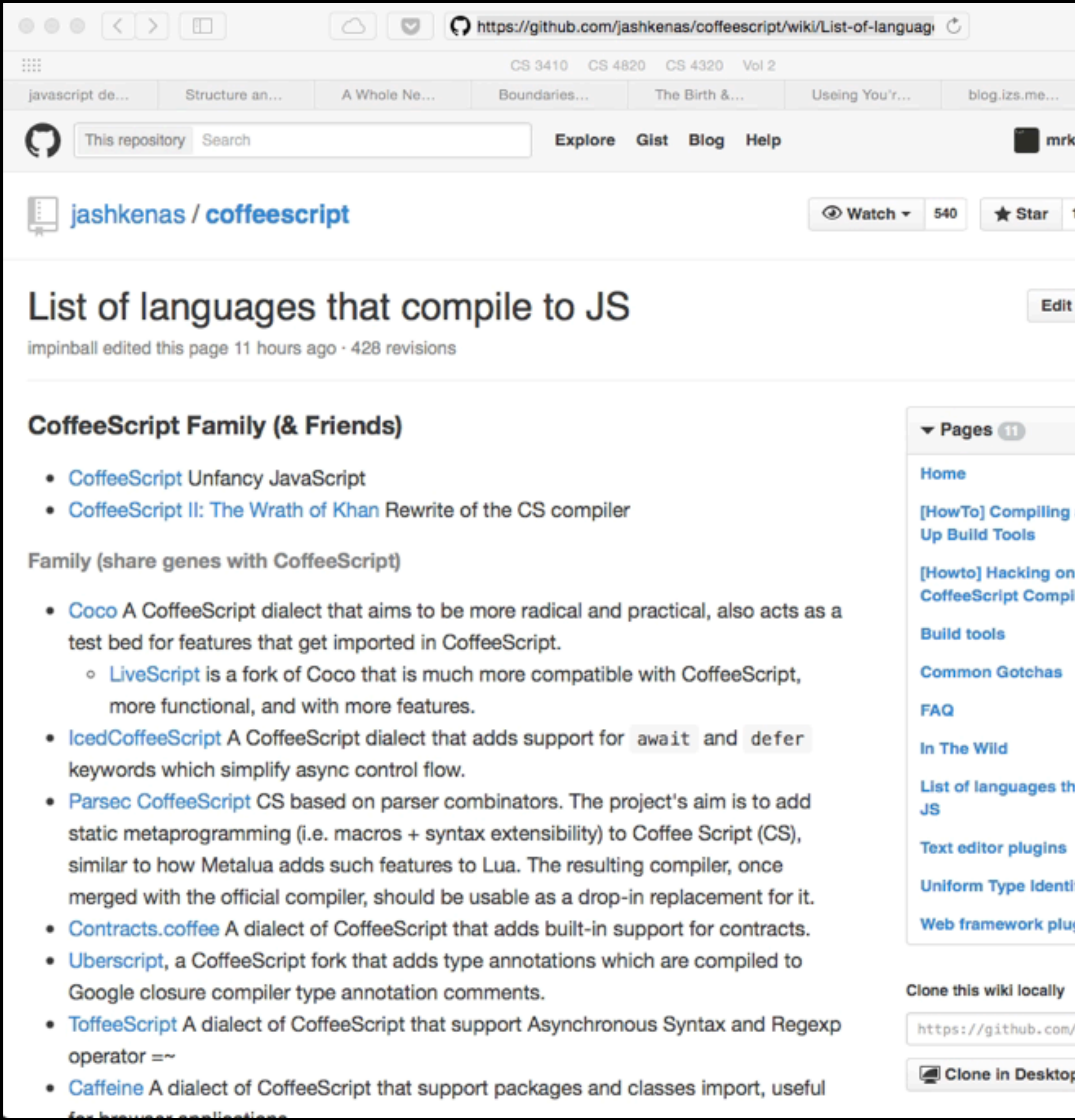
ES6

arrows
classes
enhanced object literals
template strings
destructuring
default + rest + spread
let + const
iterators + for..of
generators
unicode
modules
module loaders
map + set + weakmap + weakset
proxies
symbols
subclassable built-ins
promises
math + number + string + object APIs
binary and octal literals
reflect api
tail calls

The Good	The Bad
Fast	
NPM	
JavaScript	JavaScript

The Good	The Bad	The Ugly
Fast (out of the box)		
NPM		
		JavaScript

https://github.com/jashkenas/coffeescript/wiki/List-of-languages-that-compile-to-JS



List of languages that compile to JS

impinball edited this page 11 hours ago · 428 revisions

CoffeeScript Family (& Friends)

- CoffeeScript Unfancy JavaScript
- CoffeeScript II: The Wrath of Khan Rewrite of the CS compiler

Family (share genes with CoffeeScript)

- Coco A CoffeeScript dialect that aims to be more radical and practical, also acts as a test bed for features that get imported in CoffeeScript.
  - LiveScript is a fork of Coco that is much more compatible with CoffeeScript, more functional, and with more features.
- IcedCoffeeScript A CoffeeScript dialect that adds support for `await` and `defer` keywords which simplify async control flow.
- Parsec CoffeeScript CS based on parser combinators. The project's aim is to add static metaprogramming (i.e. macros + syntax extensibility) to Coffee Script (CS), similar to how Metalua adds such features to Lua. The resulting compiler, once merged with the official compiler, should be usable as a drop-in replacement for it.
- Contracts.coffee A dialect of CoffeeScript that adds built-in support for contracts.
- Uberscript, a CoffeeScript fork that adds type annotations which are compiled to Google closure compiler type annotation comments.
- ToffeeScript A dialect of CoffeeScript that support Asynchronous Syntax and Regexp operator =~
- Caffeine A dialect of CoffeeScript that support packages and classes import, useful

ASM