Our last conversation got me thinking about programming with Unicode characters, which in turn got me thinking about what languages use Unicode on something deeper than a "we support UTF-8" level. For example, in Scala [1] (and APL preceding it [2], though not with Unicode), you can

```
def V(b:LogicBoolean)=new LogicBoolean(this.value || b.value)
def Λ(b:LogicBoolean)=new LogicBoolean(this.value && b.value)
def →(b:LogicBoolean)=(¬(this)) V (this Λ b)
object LogicBoolean {
def ¬(a:LogicBoolean)=new LogicBoolean(!a.value)
implicit def boolToLogicBool(x:Boolean)=new LogicBoolean(x)
}
```

and thus create a nice logic DSL

```
def expr(a:Boolean,b:Boolean,c:Boolean,d:Boolean)=
( (a Λ b) V ¬(a Λ c) ) Λ ((c V d) Λ ¬(b) )
```

[3]

But what about something even deeper than Unicode-for-prexisting-DSL?
And what would a language that worked exclusively on Unicode look like?

Emojinal [4] is an emoji-only, stack-based programming language. It transpiles to JavaScript [12], and runs on node [5]. It was first released late last year [6]. It's creator has really interesting ideas about symbols, programming language, culture, and bias [7]. Stack-based languages are weird, especially higher-level ones, compared to the more conceptually dominant register-based machines, though we run into them in unexpected places [8].

Let's print hello world:

```
imperative pseudocode: println("hello world");
forth:                 : HELLO ."Hello World " ;
ascii pseudocode:      ( . ":wave: :earth_americas:" )
emojinal:              🚀 🍕 ✌👋🌎✌ 🚀
```

Syntax:
Prefix notation. Emoji codepoints (in the Unicode meaning) map to a stack-based pseudocode, which is parsed into JavaScript. There are Emojinal tokens that map directly to useful JavaScript tokens [13].

Compilation:
The core compiling loop takes emoji characters, converts them to ASCII pseudocode, parses that pseudocode (using a library called PEG [17]) which results in a Forth-like JavaScript language (called FJS) then sends the FJS code to the FJS interpreter to execute.

Execution:

Execution is done by the FJS interpreter, and uses a single stack for keeping state. FJS executes on node.js. Basically, it's turtles all the way down.

Emojinal. Write in Emoji, transpile into JavaScript, run anywhere :)
Unfortunately, JavaScript has a Unicode Problem [10]!
Some Unicode is simple to deal with in JS:

```
>> 'I \u2661 JavaScript!'
'I ♡ JavaScript!'
```

[10]

But Emoji is on an Unicode "astral plane", needing more than 4 hex digits to represent its code points. In EMCAScript 6 we will get Unicode code point escapes:

```
>> '\u{1F4A9}'
'💩' // U+1F4A9 PILE OF POO
```

for up to 6 hex digits, thus able to address all the astral planes. But for ES5, we have to use a "surrogate pair," a pair of 4 hex digits:

```
>> '\uD83D\uDCA9'
'💩' // U+1F4A9 PILE OF POO
```

which, if you notice, does not map directly to the actual 6 digit hex code point. So there's a translation table between each surrogate pair and code point. And basically everything related to string of Emoji in JavaScript will be weird.

Just be grateful we aren't talking about a programming language in Tamil [11].

Back to Emojinal.

Only on FF on Mac. Or Safari on iOS.

Let's dive in with an analysis of the the example code running on a heroku instance, that demonstrates how to create a DSL for Processing, the JavaScript graphics library, in Emojinal [5]. Here's [14] how Emojinal can work as a DSL.

Demo emojinal.herokuapp.com

Very nice, I think. With a DSL, you can start to see the power of how Emojinal can take advantage of the semantics we find in emoji to compound expressiveness while still retaining conciseness and readability. It's actually really interesting to just glance down the source code listing for this example, as the different colors and symbols really give rhythm to the listing. I dare say there's no need for syntax highlighting since the emoji create enough similar/difference contrast to remove the need for additional meta-lexical meaning.

wandering snake line:

🎨🔗➡⬆⌚➡️⌚⬆️

wandering pile of Poo:

🎨✏️✌️💩✌️⌚➡️⌚⬆️

Weird—natural environment to work with this is Safari on iOS. Why? Screen size more natural, keyboard is virtual (and can switch to emoji, Recent tab, etc), cloud app. Emoji is the first mobile-native alphabet/keyboard. Emoji grew from mobile devices, so it makes sense that the natural way to edit an emoji programming language is mobile.

Sapir-Worf hypothesis [18]. Languages limit how you think. Programming languages limit how you think. Giving you a new language with a crazy set of novel tokens expands your mind.

Let's talk about using just emoji, again. Where do you mostly use emoji, if at all. Mobile. Emoji was born in mobile (NTT, 1999). It's the first language to be mobile-first. And, if you just look at the physical act of typing (put aside debugger concerns, etc), emoji fit most naturally on an on-screen virtual keyboard (or really, a sample pad/soundboard type thing). Addie Wagenknecht [15] & Ramsey Nasser [16] are very aware that they made a language that is fundamentally compatible with authoring on mobile devices in a way that ASCII is not, and that is culturally portable in a way that an ASCII or even English semantic meaning is not. Bring on the 860-character emoji future! Or, in emoji, 👋👯📱➡SOON

[1] http://gabrielsw.blogspot.com/2009/06/boolean-algebra-internal-dsl-in-scala.html
[2] http://en.wikipedia.org/wiki/APL_%28programming_language%29
[3] http://gabrielsw.blogspot.com/2009/06/boolean-algebra-internal-dsl-in-scala.html
[4] https://github.com/wheresaddie/Emojinal
[5] http://emojinal.herokuapp.com/
[6] https://github.com/wheresaddie/Emojinal/commits/master
[7] http://www.fastcolabs.com/3024664/under-the-hood-of-the-all-emoji-programming-language
[8] http://c2.com/cgi/wiki?StackBasedLanguage
[9] GOAT programming book cover (Forth, of course): http://www.globalnerdy.com/2007/09/14/reimagining-programming-book-covers/
[10] http://mathiasbynens.be/notes/javascript-unicode
[11] http://mathiasbynens.be/notes/javascript-unicode#other-grapheme-clusters
[12] http://www.stevefenton.co.uk/Content/Blog/Date/201211/Blog/Compiling-Vs-Transpiling/
[13] https://github.com/wheresaddie/Emojinal/blob/master/keywords
[14] https://github.com/wheresaddie/Emojinal/blob/master/emojinal/graphics.js
[15] https://twitter.com/wheresaddie
[16] http://nas.sr/ & https://twitter.com/ra
[17] http://pegjs.majda.cz
[18] http://en.wikipedia.org/wiki/Linguistic_relativity