Esoteric Programming Languages

Jason Bock

Developer Advocate

Rocket Mortgage

Personal Info

- http://www.jasonbock.net
- https://mstdn.social/@jasonbock
- https://www.github.com/jasonbock
- https://www.youtube.com/JasonBock
- jason.r.bock@outlook.com

Downloads

https://github.com/JasonBock/IronBefunge
https://github.com/JasonBock/WSharp
https://github.com/JasonBock/Presentations

There's one language in this talk that will be censored, but…it has a "bad" word in it. So … just so you know ☺.

If that word offends you, just think of cute kittens.

https://unsplash.com/photos/vmFEBIEz0hQ

Trying to define what "esoteric" is, is not an easy endeavor as it's somewhat subjective. Is that esoteric art? Why or why not?

https://www.sfgate.com/art/article/Esoteric-Street-art-at-Space-Gallery-3177167.php

esoteric [es-*uh*-**ter**-ik] SHOW IPA 🔊 ☆

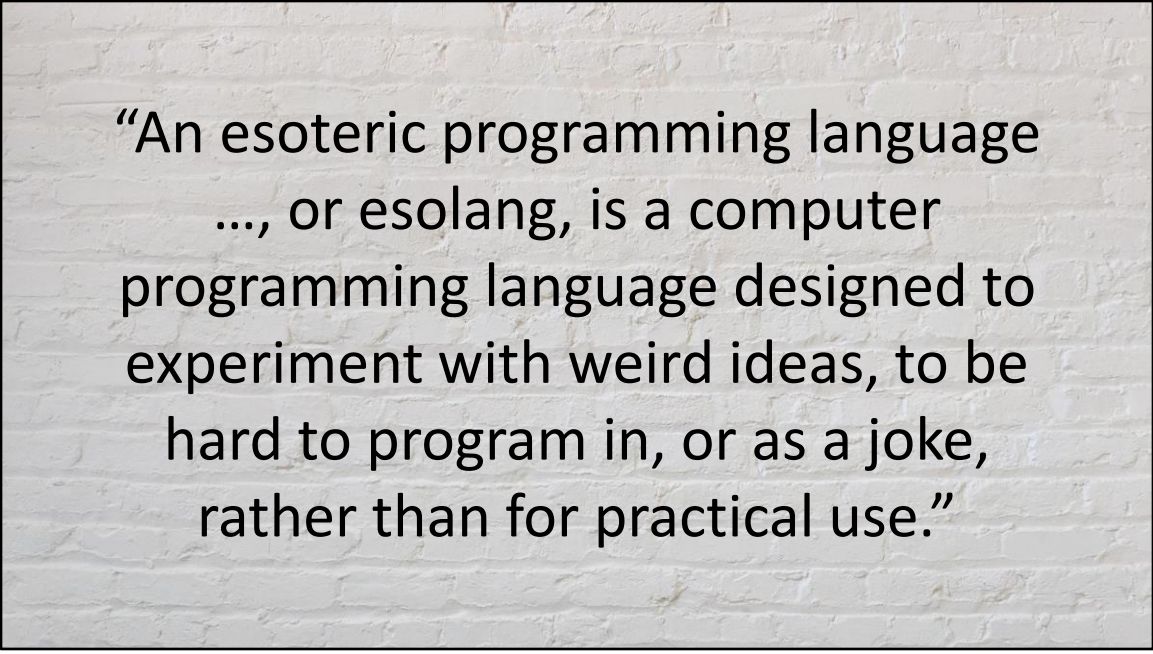See synonyms for *esoteric* on Thesaurus.com   🟧 Middle School Level

adjective

1 understood by or meant for only the select few who have special knowledge or interest;
   recondite:
   poetry full of esoteric allusions.

2 belonging to the select few.

3 private; secret; confidential.

4 (of a philosophical doctrine or the like) intended to be revealed only to the initiates of a
   group:
   the esoteric doctrines of Pythagoras.

Here's an "official" definition from dictionary.com. Does this help?

https://www.dictionary.com/browse/esoteric

> "An esoteric programming language …, or esolang, is a computer programming language designed to experiment with weird ideas, to be hard to program in, or as a joke, rather than for practical use."

What's the point of spending any time exploring esoteric languages? Well, it's fun. It's also artistic in a way. It's a way to just do something different in the world of software development.

https://esolangs.org/wiki/Esoteric_programming_language
https://unsplash.com/photos/4Zaq5xY5M_c
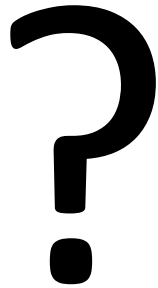
So, why spend time with these esoteric languages?

They're funny. They're meant to be a joke, and while they can be creative, they also make me laugh ☺

https://unsplash.com/photos/X1GZqv-F7Tw

They can also be educational. They're typically not that difficult to write a compiler or interpreter for, so if you want to learn that aspect of software development, pick one of them.

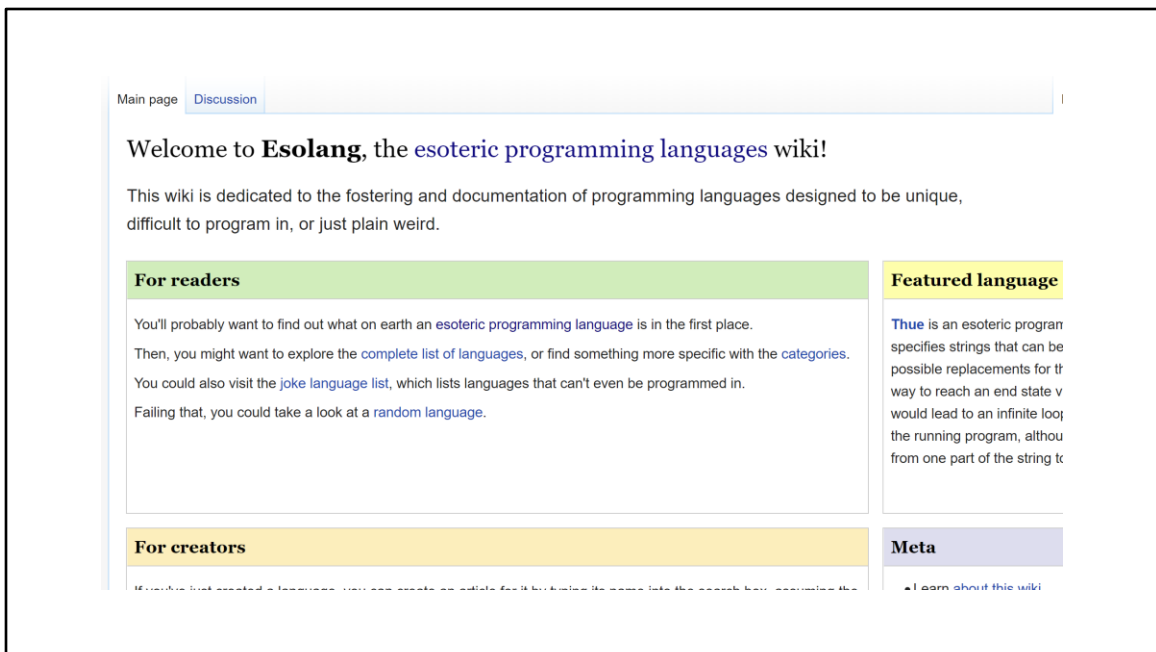https://www.pexels.com/photo/books-in-black-wooden-book-shelf-159711/

If you dive into the world of EPLs, beware. Documentation is light or non-existent, compilers/interpreters may not work as expected. You don't get a lot of direction, and sometimes you just have to use your best judgement.

https://www.pexels.com/photo/art-background-brick-brick-texture-272254/

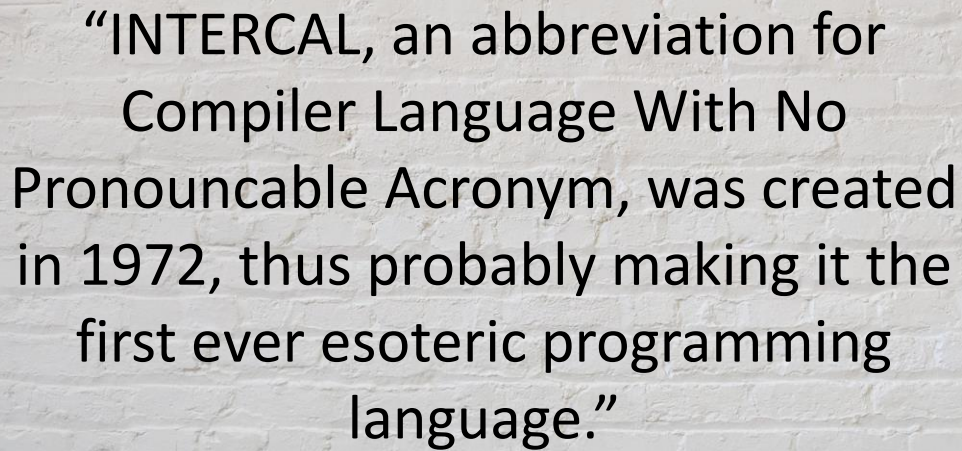Here's the best resource to find out information on everything esoteric programming language related.

https://esolangs.org/wiki/Main_Page

Let's start with INTERCAL. If you get the Monty Python reference, that'll help with understanding the sense of humor behind this language.

https://mlpnk72yciwc.i.optimole.com/w:1920/h:1080/q:auto/https://www.bleedingcool.com/wp-content/uploads/2019/10/monty-python-bbc.jpg

"INTERCAL, an abbreviation for Compiler Language With No Pronouncable Acronym, was created in 1972, thus probably making it the first ever esoteric programming language."

I love that the acronym and the words behind it do not line up at all.

https://unsplash.com/photos/4Zaq5xY5M_c

DON'T
DO NOT
PLEASE DON'T
PLEASE DO NOT
COME FROM
ABSTAIN

1/3 to 1/5 of the commands must be polite

http://esolangs.org/wiki/INTERCAL

E632 - THE NEXT STACK RUPTURES. ALL DIE. OH, THE EMBARRASSMENT!

E990 - YOU HAVE TOO MUCH ROPE TO HANG YOURSELF

E333 - YOU CAN'T HAVE EVERYTHING, WHERE WOULD YOU PUT IT?

E139 - I WASN'T PLANNING TO GO THERE ANYWAY

E774 - RANDOM COMPILER BUG

The error messages are priceless

http://www.catb.org/~esr/intercal/ick.htm#Errors

```
--------------------------------------------------------------->8---

           PLEASE NOTE THAT THIS MAY ONLY RUN ON C-INTERCAL

           PLEASE DO ,1 <- #13
           DO ,1 SUB #1 <- #238
           DO ,1 SUB #2 <- #112
           DO ,1 SUB #3 <- #112
           DO ,1 SUB #4 <- #0
           DO ,1 SUB #5 <- #64
           DO ,1 SUB #6 <- #238
           DO ,1 SUB #7 <- #26
           DO ,1 SUB #8 <- #248
           DO ,1 SUB #9 <- #168
           DO ,1 SUB #10 <- #24
           DO ,1 SUB #11 <- #16
           DO ,1 SUB #12 <- #158
           DO ,1 SUB #13 <- #52

           PLEASE READ OUT ,1
           PLEASE GIVE UP

--------------------------------------------------------------->8---
```

I'll trust that this actually does what it's supposed to do

http://esolangs.org/wiki/INTERCAL

Brainf**k is a language that comes across as being "complicated", but as you'll see, it's actually quite simple.

https://www.pexels.com/photo/assorted-puzzle-game-1586951/

Commands	Description
>	Move pointer to the right
<	Move pointer to the left
+	Increment the memory cell under the pointer
-	Decrement the memory cell under the pointer
.	Output the character signified by the cell at the pointer
,	Input a character and store it in the cell at the pointer
[	Jump past the matching] if the cell under the pointer is 0
]	Jump back to the matching [if the cell under the pointer is nonzero

BF works on a tape machine as storage. The tape has individual cells with integer values, and the commands work on the current cell's value.

++++++++[>+++++++++<-]>.

8 9
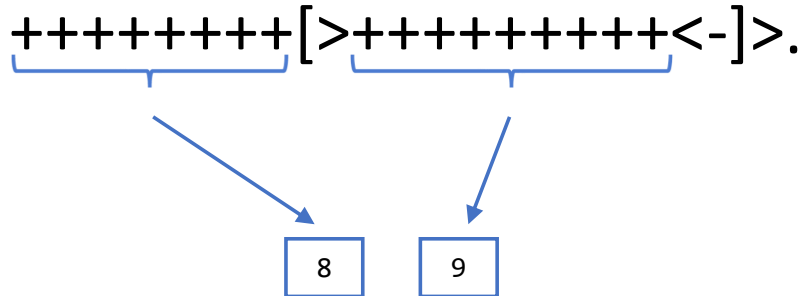
8 * 9 = 72 => 'H'

This code prints "H", which could be the start of "Hello World"

All the plusses put 8 into the current cell.
Then "[" doesn't jump to "]" because we're at 8.
We move to the next cell with ">", and put 9 in that cell.
We move back to the first cell with "<", minus one, and we jump back to "]" because 7 isn't non-zero.
We move to the 9 cell, and add 9 more.
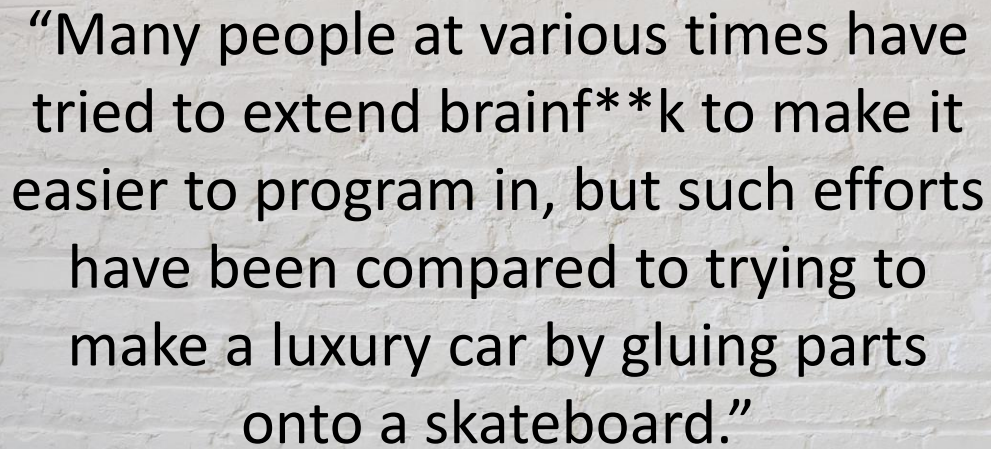We keep looping until the 8 cell goes down to zero, which causes "]" to move on.
">" moves us to the "72" cell, and we print 'H' via "."

```
++++++++[>++++[>++>+++
>+++>+<<<<-]>+>+>-
>>+[<]<-]>>.>---
.+++++++..+++.>>.<-
.<.+++.------.--------
.>>+.>++.
```
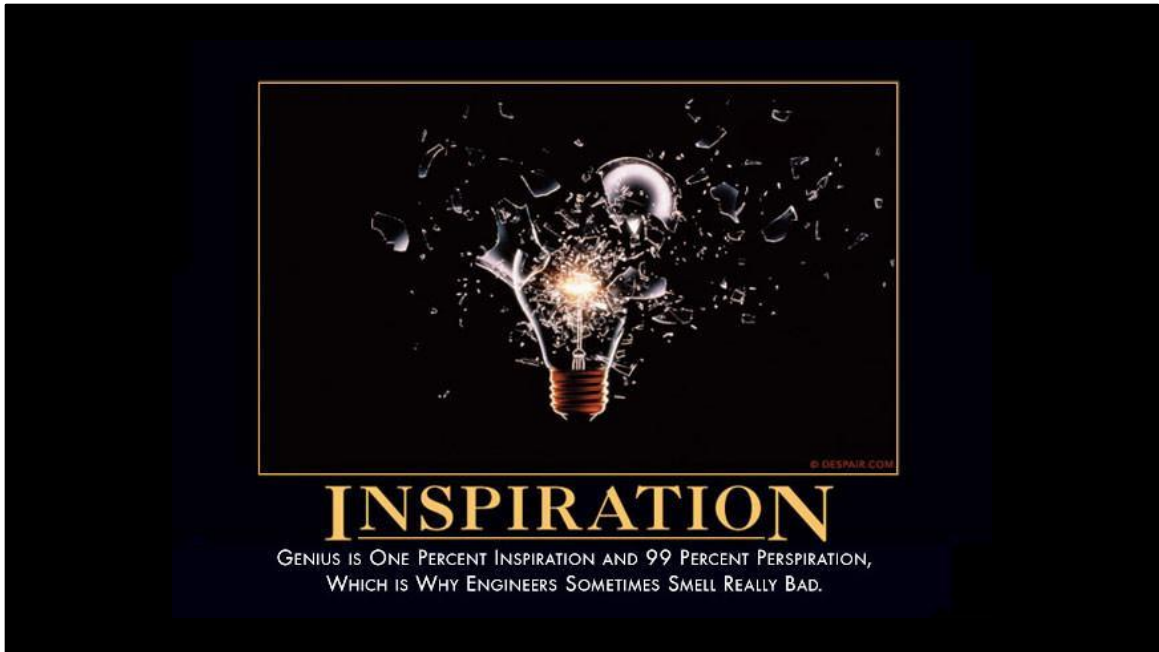
Here's "Hello world" in BF in its' entirety.

> "Many people at various times have tried to extend brainf**k to make it easier to program in, but such efforts have been compared to trying to make a luxury car by gluing parts onto a skateboard."

There's been work to extend BF, and I love this quote.

https://unsplash.com/photos/4Zaq5xY5M_c

One interesting aspect to these esoteric languages is they're good places to start to learn how to write your own compilers and interpreters (something I'm a newbie at myself). They can inspire you to explore programming by building these assets to get code written in a language.

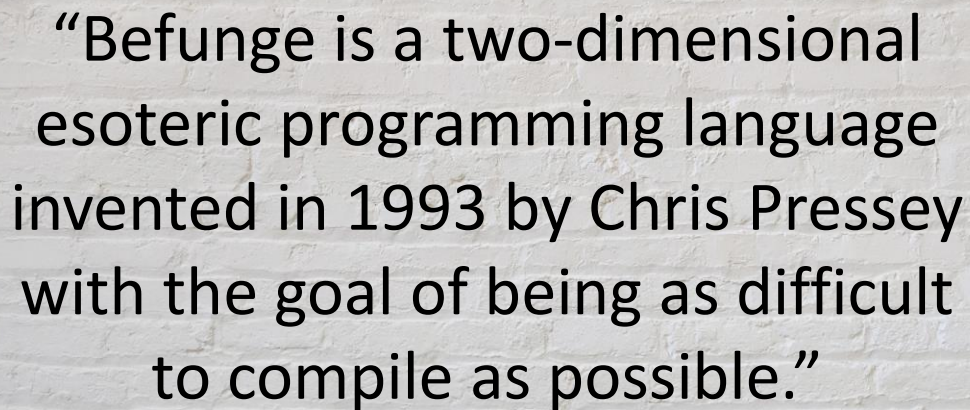https://despair.com/products/inspiration
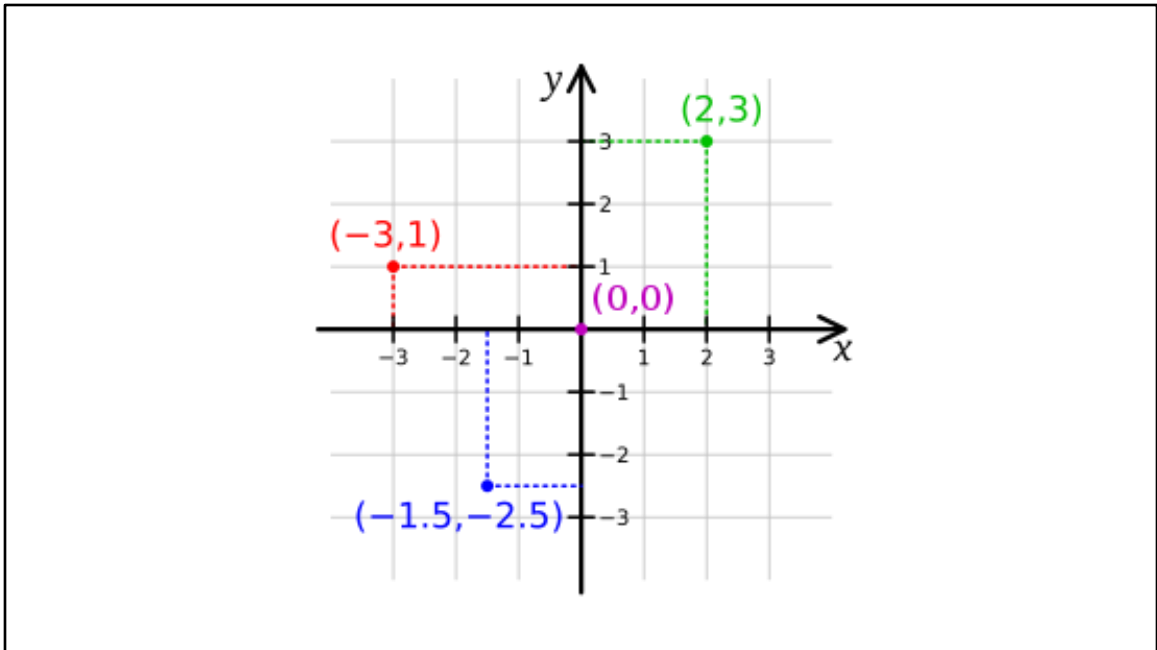
Befunge is an interesting language. It's cell-based in 2D, so you move around, interpret the character at that cell, and do some kind of activity based on the value.

https://upload.wikimedia.org/wikipedia/commons/thumb/0/0e/Cartesian-coordinate-system.svg/300px-Cartesian-coordinate-system.svg.png

"Befunge is a two-dimensional esoteric programming language invented in 1993 by Chris Pressey with the goal of being as difficult to compile as possible."

It's almost impossible to compile Befunge code, but you can write an interpreter.

https://unsplash.com/photos/4Zaq5xY5M_c
http://www.esolangs.org/wiki/Befunge

Instruction(s)	Description
>, V, <, ^	Change direction
"	Toggle string mode
0 – 9	Push value on to stack
:	Duplicate top stack value
_	Horizontal IF
,	Output stack value as ASCII
*	Multiply two top stack values
@	End program

Here are some of the instructions

```
v"Canonical hello world"
>                        v
>v"Hello world!"0<
,:
^_25*,@
```

Here's "Hello World" in Befunge

Compilation

As stated, the design goal for Befunge was to create a language which was difficult to compile. This was realized by two main features:

- self-modifying – the `p` instruction can write new instructions into the playfield; and
- multi-dimensional – the same instruction can be executed in four different contexts (in a left-to-right series of instructions, or right-to-left, or upward or downward.)

Nevertheless, these obstacles have been overcome to some degree, and Befunge compilers have been written, using appropriate techniques.

Explanation of why compilation is hard. The

https://esolangs.org/wiki/Befunge#Compilation

Demo: IronBefunge

Esoteric Programming Languages

Let's take a look at IronBefunge

Whenever is a lazy language. It's akin to sitting in a comfortable chair, and just randomly flipping channels on the TV, except in Whenever, you're randomly executing code.
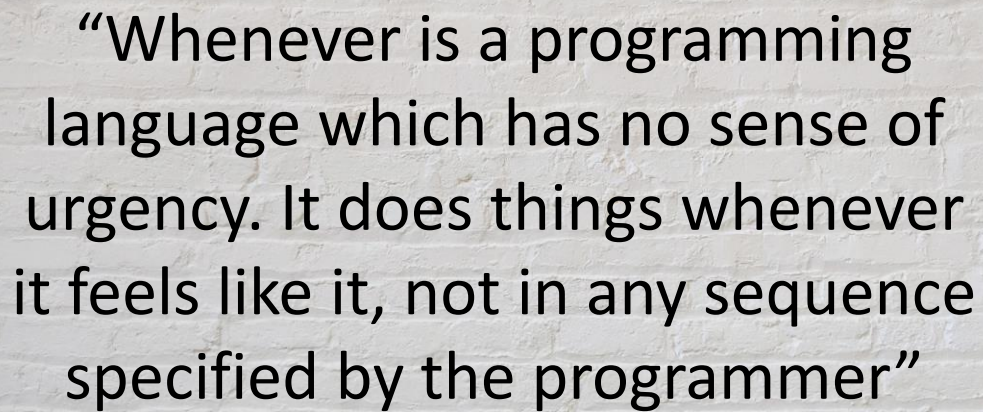
https://unsplash.com/photos/imAfCYq7KH0

Maybe a better definition is that it's a random language. Statements are run randomly, so an application's output can change each time it executes.

https://unsplash.com/photos/hFSnuPi0t3Q

"Whenever is a programming language which has no sense of urgency. It does things whenever it feels like it, not in any sequence specified by the programmer"

http://www.dangermouse.net/esoteric/whenever.html

- Program code lines will always be executed, eventually (unless we decide we don't want them to be), but the order in which they are executed need not bear any resemblance to the order in which they are specified.
- Variables? We don't even have flow control, we don't need no steenking variables!
- Data structures? You have got to be kidding.

There are no variables, data structures, etc. There's just statements with a few control structures in place. That's it.

http://www.dangermouse.net/esoteric/whenever.html

Essentially every line of code is thrown into a "grab bag" and executed at random.
Each line of code also has a count associated with it, we'll get to that in a moment.

- *defer* – defers execution of a statement
- *again* – keeps a statement in the to-do list after execution
- *N(x)* – returns the number of lines specified by x
- *U(x)* – returns the Unicode representation of x

There are a small handful of functions that help with bringing some sense to Whenever.

```
1 defer (4 || N(1) < N(2) && N(2) < N(3))
print(N(1) + " bottles of beer on the wall, "
+ N(1) + " bottles of beer,");

2 defer (4 || N(1)==N(2)) print("Take one down
and pass it around,");

3 defer (4 || N(2)==N(3)) print(N(1) + "
bottles of beer on the wall.");

4 1#98,2#98,3#98;
```

For example, this program will do 99 bottles of beer on the wall. The defer function won't let 1, 2, or 3 run until 4 runs, and then the other half ensures 1, then 2, then 3 run 99 times

http://www.dangermouse.net/esoteric/whenever.html

```
1 again (1) defer (3 || N(1)<=N(2) || N(7)>99) 2#N(1),3,7;
2 again (2) defer (3 || N(2)<=N(1) || N(7)>99) 1#N(2),3,7;
3 defer (5) print(N(1)+N(2));
4 defer (5) print("1");
5 4,-3,7;
6 defer (4) 3;
7 7;
8 defer (N(7)<100) -1#N(1),-2#N(2),-7#100,-3;
9 defer (3 || 6) 1,3;
```

Feel free to follow the logic in this one ☺. It's the Fibonacci numbers (first 100)

Demo: WSharp

Esoteric Programming Languages

Let's take a look at WSharp

At the end, this may all feel like a waste of time.

http://ecoopportunity.net/wp-content/uploads/2011/06/wasting-time.jpg

But if YOU don't feel like it was time wasted (e.g. you got something out of it) then it's not wasted time.

http://images.piccsy.com/cache/images/time-you-enjoy-wasting-is-not-wasted-time-96489-500-500.jpg

Esoteric Programming Languages

Jason Bock

Developer Advocate

Rocket Mortgage

Remember…
- https://github.com/JasonBock/IronBefunge
- https://github.com/JasonBock/WSharp
- https://github.com/JasonBock/Presentations
- Links to code and slide deck
- References in the notes on this slide

http://www.esolangs.org/wiki/Main_Page
https://devblogs.microsoft.com/visualstudio/how-to-debug-and-profile-any-exe-with-visual-studio/