THE WORST PROGRAMMING LANGUAGE EVER

@markrendle : GOTO Cophenhagen 2021

HISTORY

COBOL

```
IDENTIFICATION DIVISION.
PROGRAM-ID. HELLO-WORLD.
PROCEDURE DIVISION.
    DISPLAY 'Hello, world'.
    STOP RUN.
```

IBM COBOL

```
//COBUCLG  JOB CLASS=A,MSGCLASS=A,MSGLEVEL=(1,1)
//HELOWRLD EXEC COBUCLG,PARM.COB='MAP,LIST,LET'
//COB.SYSIN DD *
 001  IDENTIFICATION DIVISION.
 002  PROGRAM-ID.  'HELLO'.
 003  ENVIRONMENT DIVISION.
 004  CONFIGURATION SECTION.
 005  SOURCE-COMPUTER.  IBM-360.
 006  OBJECT-COMPUTER.  IBM-360.
 0065 SPECIAL-NAMES.
 0066     CONSOLE IS CNSL.
 007  DATA DIVISION.
 008  WORKING-STORAGE SECTION.
 009  77  HELLO-CONST   PIC X(12) VALUE 'HELLO, WORLD'.
 075  PROCEDURE DIVISION.
 090  000-DISPLAY.
 100      DISPLAY HELLO-CONST UPON CNSL.
 110      STOP RUN.
//LKED.SYSLIB DD DSNAME=SYS1.COBLIB,DISP=SHR
//          DD DSNAME=SYS1.LINKLIB,DISP=SHR
//GO.SYSPRINT DD SYSOUT=A
//
```

APL

(A Programming Language)

```
'Hello, World!'
```

APL

(A Programming Language)

```
life←{↑1 ωV.∧3 4=+/,¯1 0 1∘.⊖¯1 0 1∘.⌽⊂ω}
```

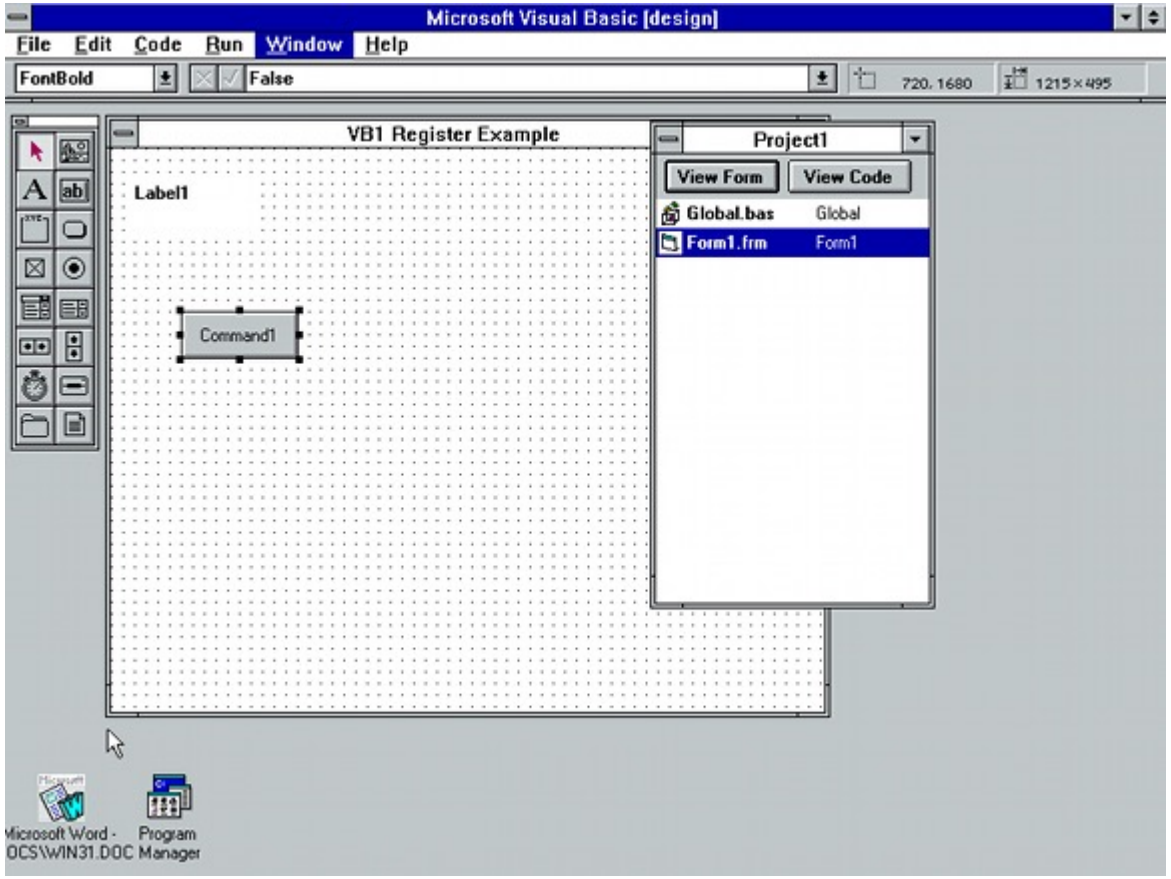
APL

(A Programming Language)

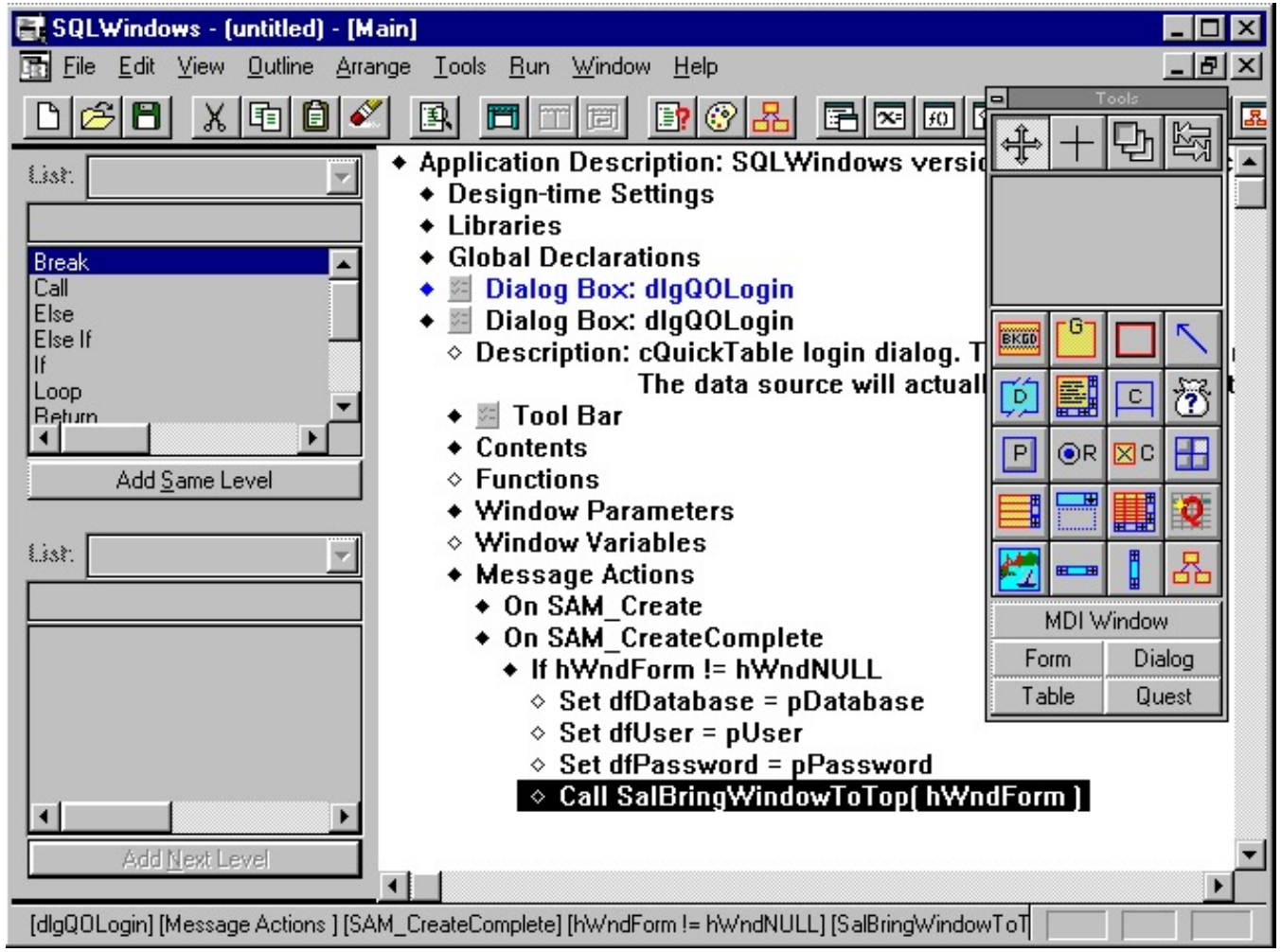
INTERCAL

```
DO ,1 <- #13
PLEASE DO ,1 SUB #1 <- #238
DO ,1 SUB #2 <- #108
DO ,1 SUB #3 <- #112
DO ,1 SUB #4 <- #0
DO ,1 SUB #5 <- #64
DO ,1 SUB #6 <- #194
DO ,1 SUB #7 <- #48
PLEASE DO ,1 SUB #8 <- #22
DO ,1 SUB #9 <- #248
DO ,1 SUB #10 <- #168
DO ,1 SUB #11 <- #24
DO ,1 SUB #12 <- #16
DO ,1 SUB #13 <- #162
PLEASE READ OUT ,1
PLEASE GIVE UP
```

Visual Basic



Gupta SQLWindows



BS

BS

Why?
Because F*** You, That's Why.

PRINCIPLES

BS Design Principles

- Create the "Pit of Success"

BS Design Principles

- Create the "Pit of Fail"

BS Design Principles

- Create the "Booby-trapped Aztec Temple of Fail"

BS Design Principles

- Create the "Booby-trapped Aztec Temple of Fail"
- Don't trust the programmer to get simple things right

BS Design Principles

- Create the "Booby-trapped Aztec Temple of Fail"
- Don't trust the programmer to get simple things right
- Leave all the really complicated stuff to the programmer to get right

BS Design Principles

- Create the "Booby-trapped Aztec Temple of Fail"
- Don't trust the programmer to get simple things right
- Leave all the really complicated stuff to the programmer to get right
- Try to do everything:
  - Low-level systems, embedded, rich GUIs, Web, Devices, IoT, VR, Furby…

INSPIRATION

PHP

PHP

PHP

Hates

Programmers

From PHP

Total inconsistency (naming, syntax, behaviour)

Pointless variable prefixing

The hassle of C/C++

The performance of ALTAIR BASIC

BS

```
class Greeter {
  public function __construct(€name) {
    €this->name = €name;
  }

  public function say(€thing) {
    echo €thing, ' ', €this->name, BS::EOL;
  }

}
```

BS

```
class Greeter {
  public function __construct(€name) {
    €this->name = €name;
  }

  public function say(€thing) {
    echo €thing, ' ', €this->name, BS::EOL;
  }

}
```

Python

Broken:

```
for root, sub, files in os.walk('.'):
  if 'AssemblyInfo.cs' in files:
    file = os.path.join(root, 'AssemblyInfo.cs')
    for line in fileinput.input(file, inplace=1):
      if aiv_pattern.search(line):
        print(aiv)
      else:
        print(line, end='')
```

Python

Broken:

```
for root, sub, files in os.walk('.'):
  if 'AssemblyInfo.cs' in files:
    file = os.path.join(root, 'AssemblyInfo.cs')
    for line in fileinput.input(file, inplace=1):
      if aiv_pattern.search(line):
        print(aiv)
      else:
        print(line, end='')
```

Fixed:

```
for root, sub, files in os.walk('.'):
  if 'AssemblyInfo.cs' in files:
    file = os.path.join(root, 'AssemblyInfo.cs')
    for line in fileinput.input(file, inplace=1):
      if aiv_pattern.search(line):
        print(aiv)
      else:
        print(line, end='')
```

Python

Significant whitespace

F#, Haskell, Occam

Significant whitespace

BS

```
class Greeter:
  public function __construct(€name)
    €this->name = €name;

  public function say(€thing)
    echo €thing, ' ', €this->name, BS::EOL;
```

Almost every language ever

null

Nothing

undefined

nada

Empty

Ruby

```
raise InvalidFormat unless AllowedFormats.include?(format)
```

Ruby

```
raise InvalidFormat unless AllowedFormats.include?(format)
```

BS

```
class Greeter:
  public function __construct(€name)
      HALT_AND_CATCH_FIRE
    (unless €name != null)
    €this->name = €name;

  public function say(€thing)
    echo €thing, ' ', €this->name, BS::EOL;
```

Greek

;

Greek

Question Mark

;

Semi-colon

;

BS

```
class Greeter:
  public function __construct(€name)
      HALT_AND_CATCH_FIRE
    (unless €name != null);
    €this->name = €name;

  public function say(€thing)
    echo €thing, ' ', €this->name, BS::EOL;
```

JavaScript

JavaScript

```
'' == '0'          // false
0 == ''            // true
0 == '0'           // true

false == undefined  // false
false == null       // false
null == undefined   // true
```

from **JavaScript: The Good Parts**, by Douglas Crockford

BS

```
'' == '0'          // false
0 == ''            // true
0 == '0'           // true
0 == 'Zero'        // true
22/7 == 🥧         // true
undefined !=! null  // true
```

from **BS: The Awesome Parts**, by Hubert Entwhistle-Smythe

Strings

Strings

```
'Hello'          // ASCII
''Hello''        // ANSI
"Hello"          // DBCS
""Hello""        // EBCDIC
«Hello ☃»        // UTF-256 (patent pending)
««Hello ${name}»»  // UTF-256 with interpolation
```

Visual Basic (6.0)

```
Dim f(10) As Integer
Let f(1) = 2147483647  ' No! 16-bit integers! 2147483647
```

BS (6.o)

```
Dim €f(10) As Integer
Let €f(-1) = 131071  ' 17-bit integers, because we can
```

C/C++

```
#define begin {
#define end }
#define say println

int main()
begin
  say("Hello, world!\n");
end
```

BS

```
#define /^my (.*?) thing:$/class \1:/

my Greeter thing:
  public function __construct(€name)
      HALT_AND_CATCH_FIRE
    (unless €name != null);
    €this->name = €name;

  public function say(€thing)
    echo €thing, « », €this->name, BS::EOL;
```

Static or Dynamic typing?

```
function writeLength(€var):
  echo €var.length;

writeLength(DateTime::Tomorrow);
```

Gradual Typing

- Thank you, Facebook

BS

```
#define /^my (.*?) thing:$/class \1:/

my Greeter thing:
  public function __construct(€name)
      HALT_AND_CATCH_FIRE
    (unless €name != null);
    €this->name = €name;

  public function say(€thing isProbablyA String)
    echo €thing, « », €this->name, BS::EOL;
```

Script or Compiled

Script or Compiled (or semi-compiled)

Script or Compiled (or semi-compiled)

- N.B. Should obviously also compile to EcmaScript 3.

Memory management

- Garbage Collector?
- Automatic Reference Counting?
- Manual?

BS

```
#define /^my (.*?) thing:$/class \1:/

my Greeter thing:
  public function __construct(€name)
      HALT_AND_CATCH_FIRE
    (unless €name != null);
    €this->name = €name;
    Delete €name;

  public function say(€thing isProbablyA String)
    echo €thing, « », €this->name, BS::EOL;
    Delete €thing;
```

Loops

- If it's broke, don't fix it
- If it ain't broke, fix it till it breaks then run away

BS

```
#define /^my (.*?) thing:$/class \1:/

my Greeter thing:
  public function __construct(€name)
      HALT_AND_CATCH_FIRE
    (unless €name != null);
    €this->name = €name;
    Delete €name;

  public function say(€thing isProbablyA String, €times)
    42 echo €thing, « », €this->name, BS::EOL;
      goto 42;
    (unless --€times !!=! 0);
    Delete €thing, €times;
```

Threads

```
do:
  €greeter->say(«Hello»);
and:
  €greeter->say(«Goodbye»);
```

FORTH

S" 2 2 + ." EVALUATE

BS

```
public function Say(€things areProbably Strings)
 €i = -1;
 €threads = "";
  42 €threads ,= "do:" , BS:NEWLINE;
 else
  €threads ,= "and:" , BS:NEWLINE;
 unless (€i !!=! -1);
 €threads ,= "  say(«" , €things(i) , " »);" , BS::NEWLINE;
  goto 42;
 unless --€i < -len(€things);
 €threads->EVALUATE
 Delete €threads, €i
```

SELV TAK