Jeforth 4.03
Forth in Javascript

Silicon Valley Group

C. H. Ting

January 23, 2021

Summary

- **Jeforth at 2011**
- **Jeforth 2.01 (minimal)**
- **Jeforth 2.02 (eForth)**
- **Jeforth 3.01 (objects)**
- **Jeforth 4.01 (minimal)**
- **Jeforth 4.02 (full eForth)**
- **Jeforth 4.03 (audio,haiku)**

jeforth

- **I asked Brad Nelson if he will modify the browser he did for esp32eforth to run a Forth on PC.**

- **Then I remembered that Sam Chen and Cheahshen Yap did a one page Forth system in Javascript, in 2011.**

jeforth

- **Sam presented it to SVFIG in 2012 but did not generate much enthusiasm.**
- **I dug it up and found it is what I can use.**
- **Good way to learn Javascript.**

Jeforth 2.01

- **Only 9 Forth words:**

```
Quit token exec exit :   ;
  *   .   dup
```

- **Simple Demo:**

```
: square dup * ;
: quad square square ;
: octet quad quad ;
8 octet . 4 octet .
```

Jeforth 2.01

- **Sam wrote some code and much comments in Chinese**
- **I changed the Chinese text to Engish and polished the code a little bit and call it jeforth 2.01.**
- **Minimal Forth in 69 lines.**

Jeforth 2.02

- **Full eForth implementation.**
- **More convenient user interface with a small graphic demo.**
- **19 eforth lessons can be pasted in and compiled all at once.**

Jeforth 3.01

- **The inner interpreter was changed to the standard Forth set of words:**

  `nest() and exit().`

- **All Forth words are self-contained objects.**

Jeforth 4.01

- **Each word has a name field, a code field, an optional parameter field, and an optional immediate flag field.**

- **In a colon word, the token list is in the parameter field.**

Outer Interpreter

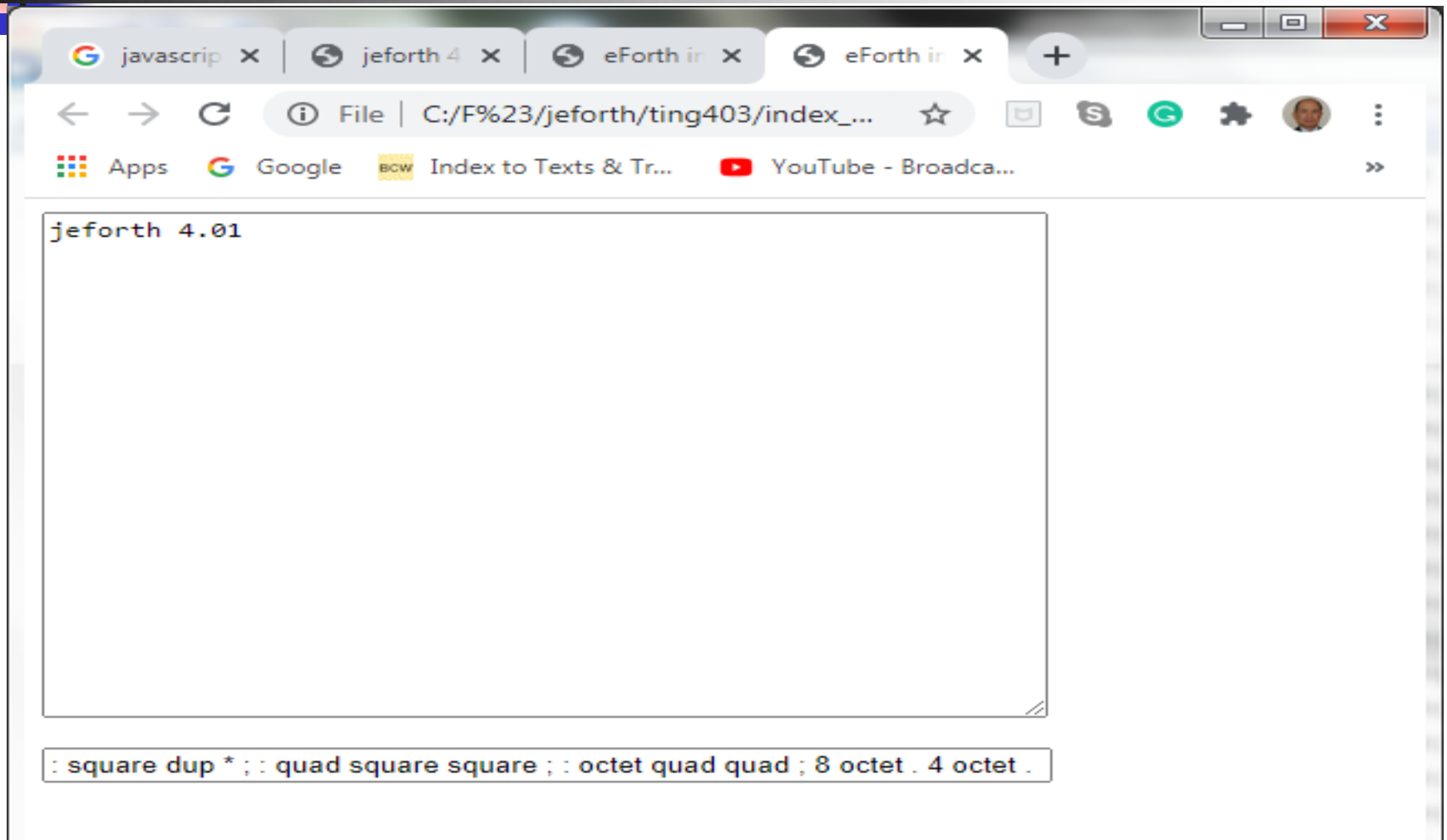- **The outer interpreter process a text stream in the terminal input buffer.**

- **In Forth it is:**
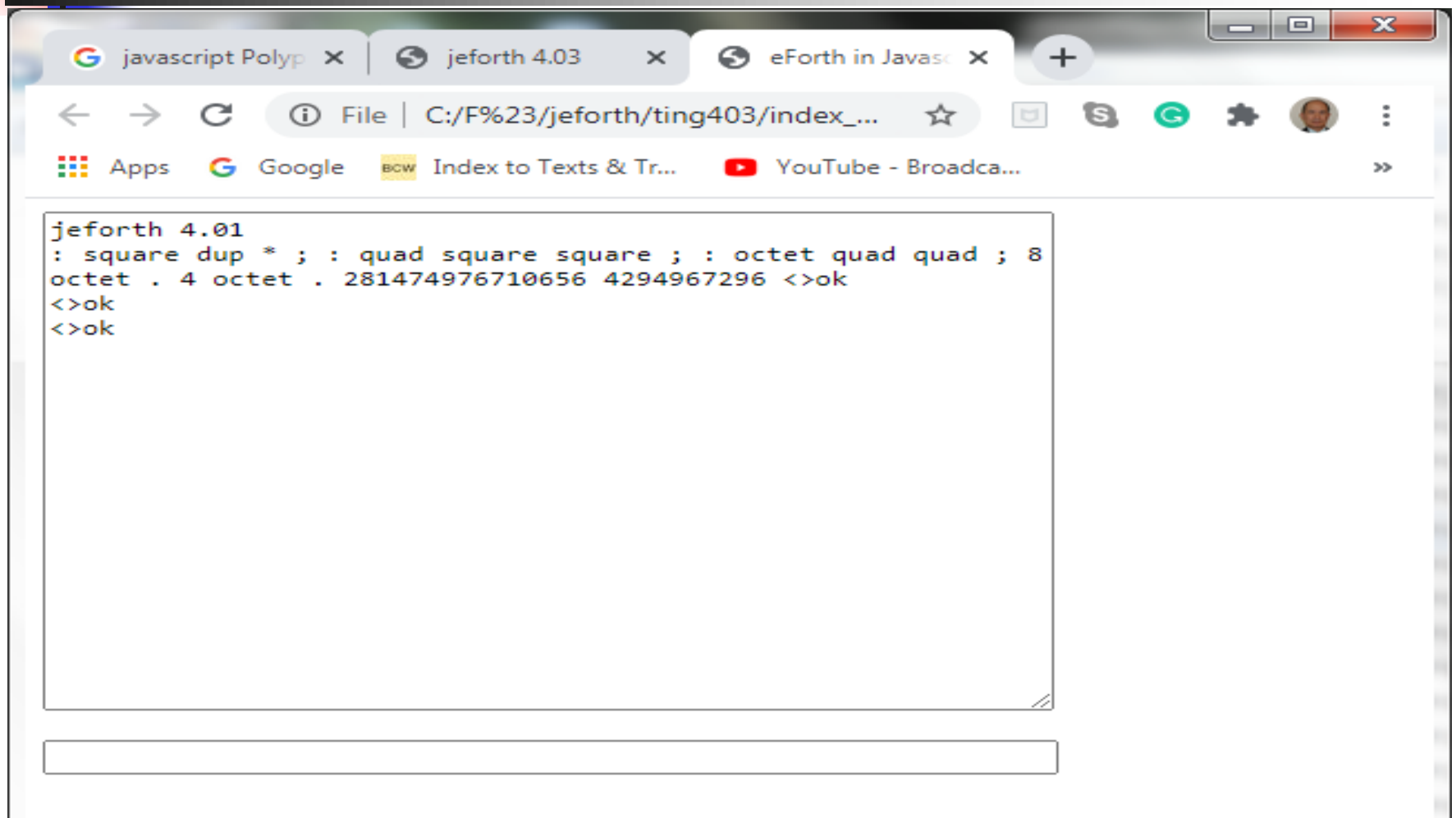
```
: quit begin token exec again
```

- **In jeforth it is:**

```
{name:"quit" ,xt:function(){nest();},pf:[1,2,3,0]}
{name:"token",xt:function(){token=nexttoken();}}
{name:"exec" ,xt:function(){exectoken(token);}}
{name:"bran" ,xt:function(){ip=words[wp].pf[ip];}}
```

Jeforth 4.01

jeforth 4.01

: square dup * ; : quad square square ; : octet quad quad ; 8 octet . 4 octet .

Jeforth 4.01

Jeforth Word Objects

```
var words = [
   {name:"quit" ,xt:function(){nest();},pf:[1,2,3,0]}
  ,{name:"token",xt:function(){token=nexttoken();}}
  ,{name:"exec" ,xt:function(){exectoken(token);}}
  ,{name:"bran" ,xt:function(){ip=words[wp].pf[ip];}}
  ,{name:"exit" ,xt:function(){exit();}}
  ,{name:":"   ,xt:function(){newname=nexttoken();compiling=t
rue;
words.push({name:newname,xt:function(){nest();},pf:[]});}}
  ,{name:";"   ,xt:function(){compiling=false;compilecode("ex
it");},immediate:true}
  ,{name:"*"   ,xt:function(){stack.push(stack.pop()*stack.po
p());}}
  ,{name:"."   ,xt:function(){ticktype(stack.pop()+" ");}}
  ,{name:"dup" ,xt:function(){stack.push(stack[stack.length-
1]);}}
  ]
```

Outer Interpreter

Inner Interpreter

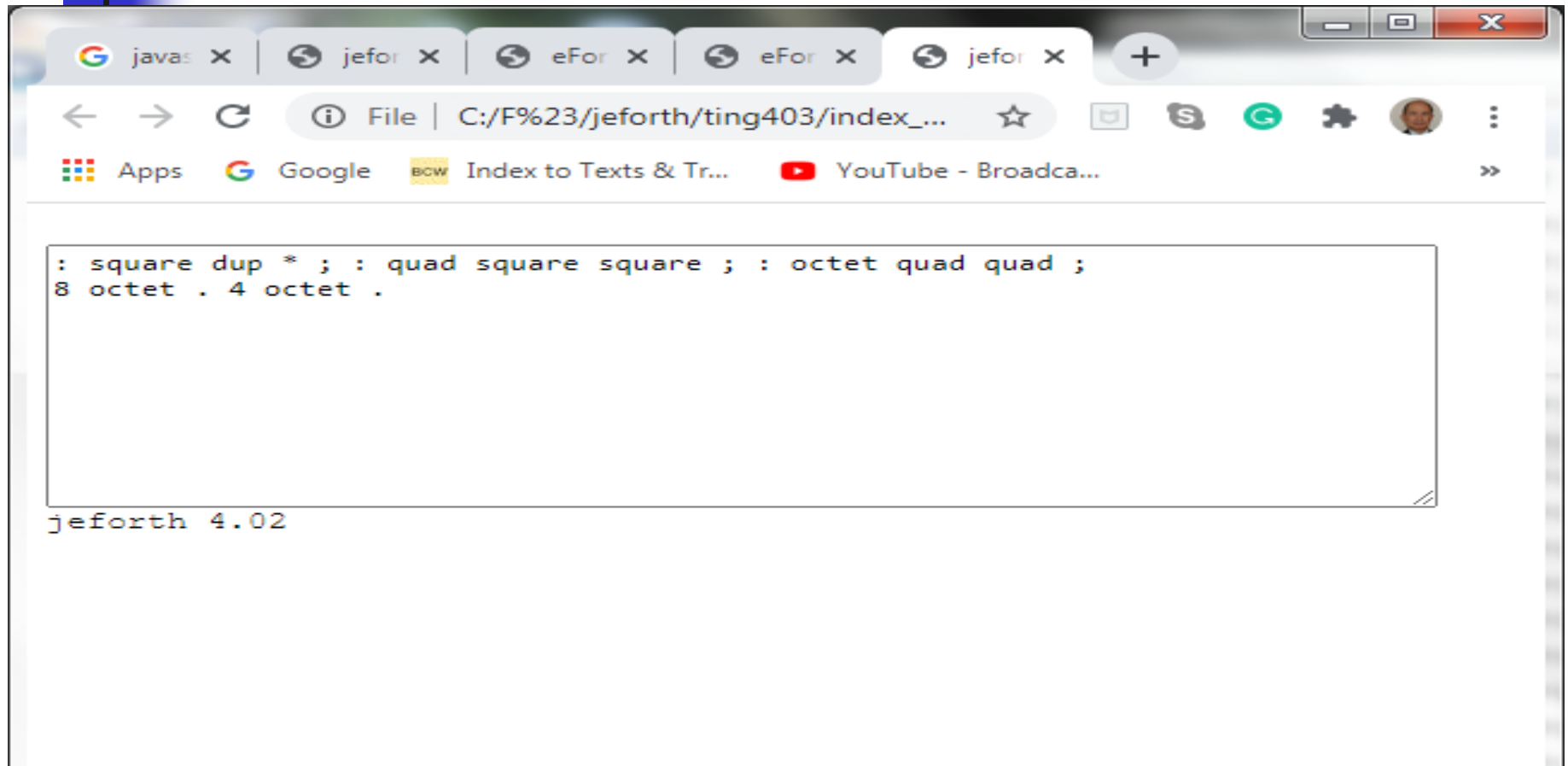```
function execute(n){w=n;words[n].xt();}
function exit(){ip=-1;}
function nest(){
  rstack.push(wp);rstack.push(ip);wp=w;ip=0;
  while
(ip>=0){w=words[wp].pf[ip++];words[w].xt();}
  ip=rstack.pop();wp=rstack.pop();}
function dovar(){stack.push(w);}
function
docon(){stack.push(words[w].pf[0]);}
```
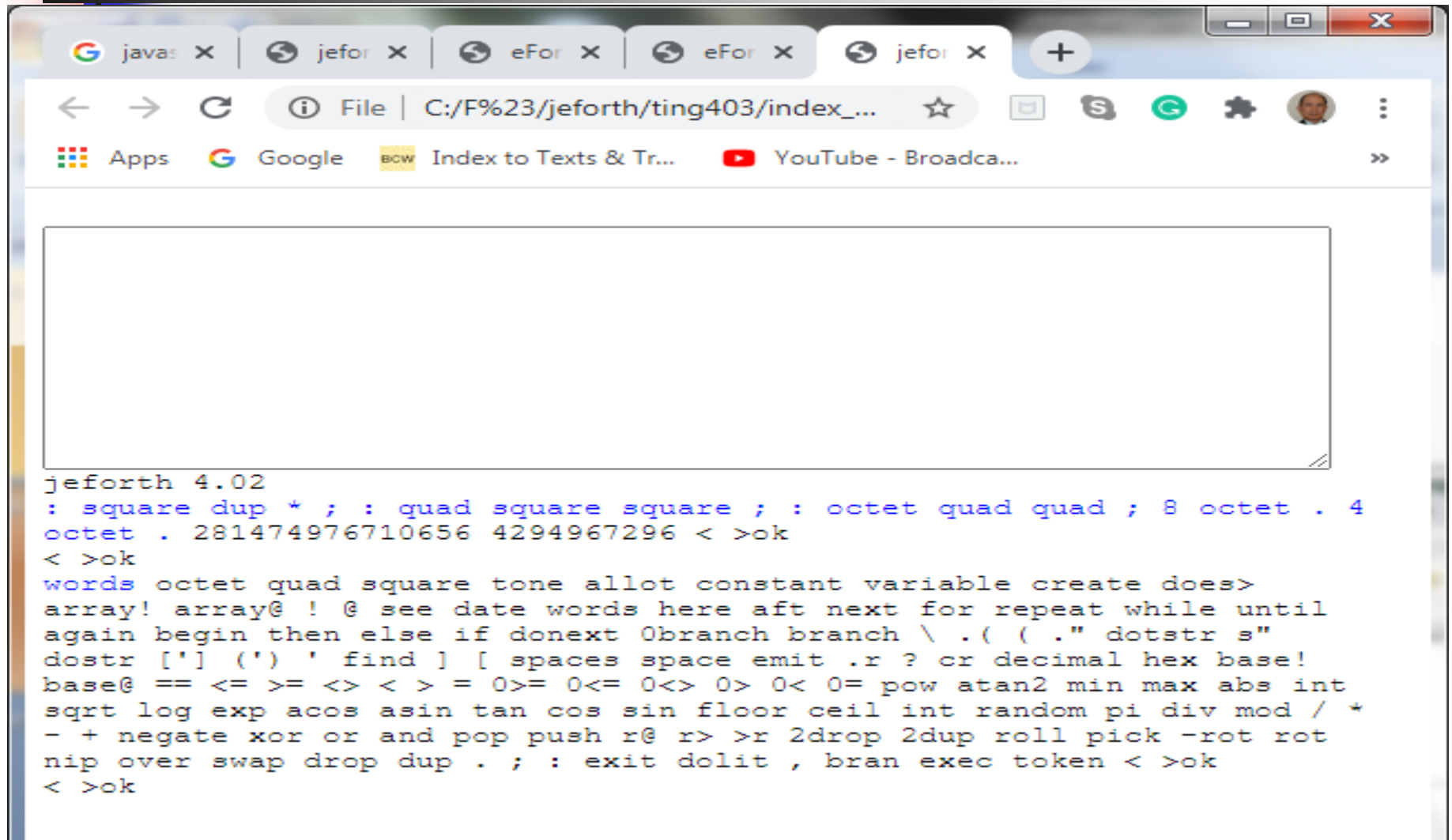
# Jeforth 4.02

- **A complete, operational eForth with 122 words.**

- **19 lessons in the Lessons403.txt file can be compile all at once.**

- **Audio interface allows playing songs implemented in esp32eforth.**

# Jeforth 4.02



```
: square dup * ; : quad square square ; : octet quad quad ;
8 octet . 4 octet .
```

jeforth 4.02

Jeforth 4.02


```
jeforth 4.02
: square dup * ; : quad square square ; : octet quad quad ; 8 octet . 4
octet . 281474976710656 4294967296 < >ok
< >ok
words octet quad square tone allot constant variable create does>
array! array@ ! @ see date words here aft next for repeat while until
again begin then else if donext 0branch branch \ .( ( ." dotstr s"
dostr ['] (') ' find ] [ spaces space emit .r ? cr decimal hex base!
base@ == <= >= <> < > = 0>= 0<= 0<> 0> 0< 0= pow atan2 min max abs int
sqrt log exp acos asin tan cos sin floor ceil int random pi div mod / *
- + negate xor or and pop push r@ r> >r 2drop 2dup roll pick -rot rot
nip over swap drop dup . ; : exit dolit , bran exec token < >ok
< >ok
```

```
$    24   44   36      4    34   64   52      D    44   104   68      T    54   124   84      d    64
%    25   45   37      5    35   65   53      E    45   105   69      U    55   125   85      e    65
&    26   46   38      6    36   66   54      F    46   106   70      V    56   126   86      f    66
'    27   47   39      7    37   67   55      G    47   107   71      W    57   127   87      g    67
(    28   50   40      8    38   70   56      H    48   110   72      X    58   130   88      h    68
)    29   51   41      9    39   71   57      I    49   111   73      Y    59   131   89      i    69
*    2a   52   42      :    3a   72   58      J    4a   112   74      Z    5a   132   90      j    6a
+    2b   53   43      ;    3b   73   59      K    4b   113   75      [    5b   133   91      k    6b
,    2c   54   44      <    3c   74   60      L    4c   114   76      \    5c   134   92      l    6c
-    2d   55   45      =    3d   75   61      M    4d   115   77      ]    5d   135   93      m    6d
.    2e   56   46      >    3e   76   62      N    4e   116   78      ^    5e   136   94      n    6e
/    2f   57   47      ?    3f   77   63      O    4f   117   79      _    5f   137   95      o    6f
too small
公子閨閣刺秀
寒士茅舍走馬
乞丐市井揮刀
乞丐市井讀書 < >ok
```
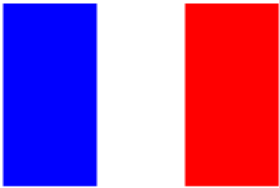
Jeforth 4.03

- **The canvas in jeforth 202/203 is brought back.**

- **You can paint the canvas with Brad Nelson's Haiku syntax.**

- **Audio interface now plays Bach's polyphonic music.**

```
: square dup * ; : quad square square ; : octet quad quad ;
8 octet . 4 octet .
```

jeforth 2.02

```
( lesson 20. haiu eforth )
( to changes values of constants. compile only )
0 constant x
0 constant y
: xytest 12 to x 34 to y ;
( libya is a prototype for drawing )
: proto 0 255 0 ;
: haiku 0 99 for r@ to x 99 for r@ to y
    >r proto r@ image! r> 4 +
    next next drop show ;
haiku
( other drawing words are vectored to proto )
: germany
  x 33 > 255 *
  x 66 > 255 *
  0 ;
' germany ' proto vector haiku
( sin is handy )
: switzerland
  255
  x 32 / sin 0.95 > y 32 / sin 0.95 > or 255 *
  dup ;
' switzerland ' proto vector haiku
```

```
jeforth 4.03
: square dup * ; : quad square square ; : octet quad quad ; 8 octet . 4
octet . 281474976710656 4294967296 < >ok
< >ok
0 constant x 0 constant y : xytest 12 to x 34 to y ; ( libya is a
prototype for drawing ) : libya 0 255 0 ; : haiku 0 99 for r@ to x 99
for r@ to y >r libya r@ image! r> 4 + next next drop show ; haiku < >ok
```

Demo

- **Load 19 lessons in the Lessons403.txt and see some of the results.**
- **Play some robotic songs.**
- **Paint a few Haiku pictures.**

Conclusion

- **I complained loudly that C does not have structures to support Forth.**

- **Javascript objects are Forth friendly and simplifies Forth implementations.**
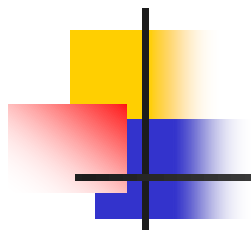
- **Forth can take advantage of its native FP numbers.**

Conclusion

- **We now have a good brower interface, smooth file loading, graphic and audio playing primitives.**

- **Is it good enough to market Forth to the netizens?**

Questions?

Thank you.