The great book of eFORTH web

version 1.0 - 22 février 2025

Autor

- Marc PETREMANN

Collaborators

- xxx

Contents

Getting started with eForth web....3
First words and first definitions....3
See other vocabularies....3
First FORTH code compilation....4
Text or graphics mode selection....4
Creating FORTH Definitions Using JavaScript....6
The combination of JavaScript and FORTH....6
Structure of a word definition with JSWORD:....6
Parameter passing petween eFORTH and JavaScript....7
Extending eFORTH's Web Vocabulary....9
Version v 7.0.7.15....10
FORTH....10
ansi....10
asm....11
internalized....11
internals....11
structures....11
tasks....11
web....11

Getting started with eForth web

First words and first definitions

If you don't have eForth web installed, you can test our online version. Link:

https://eforth.arduino-forth.com/web-eforth/index/

To see if eForth is working properly, type **words** in the eForth workspace:

Running **words** displays the contents of the **FORTH** vocabulary. All these words are equivalent to functions in other languages. FORTH is different from all other programming languages because in FORTH every new definition of a FORTH word extends the FORTH language. There is no difference between an application in FORTH and extending the dictionary.

To clear the contents of the workspace, run the word **page** .

See other vocabularies

words displays the contents of the FORTH vocabulary. But eForth web embeds other vocabularies. To see the list of these vocabularies, type **internals voclist** . Displays:

```
--> internals voclist
tasks
ansi
web
structures
internalized
internals
FORTH
ok
-->
```

Among these vocabularies, the one that will interest us is the **web vocabulary** . To see the definitions of this vocabulary, type web **vlist** . Displays:

```
--> web vlist
yielding-task yielding import rm ls download cat include-file upload
upload-file session? web-key? web-key web-type scripts scripts# random
button mouse textWidth fillText font text-size! log raw-http-upload
http-download raw-download upload-success? upload-done? upload-start
ms-ticks silence tone importScripts release keyCount getKey clearItems
removeItem getItem setItem smooth gpop gpush rotate scale translate
show-text keys-height mobile textRatios viewport@ window line fill stroke
lineTo moveTo beginPath box lineWidth color! Text gr grmode shouldEcho?
web-terminate web-key?-raw web-key-raw web-type-raw jseval JSWORD:
jsslot jseval!
ok
```

A vocabulary allows you to integrate words that can be used in a certain context. Here, the content of the **web** vocabulary is only defined for the eForth web version. The **web** vocabulary does not exist on other versions of eForth (Windows, Linux, ESP32...).

First FORTH code compilation

If a Forth definition is short, you can compile it immediately from the eForth web command prompt:

```
: myLoop 10 0 do i . loop ;
```

Then type **myLoop** , which will run the code that was just compiled by eForth web:

```
uEforth v7.0.7.9 - rev fb3db70da6d111b1fdf0
Forth dictionary: 4068200 free + 76708 used = 4144908 total (98% free)
3 x Forth stacks: 16384 bytes each
ok
-->: myLoop 10 0 do i . loop ;
ok
--> myloop
0 1 2 3 4 5 6 7 8 9 ok
-->
```

Text or graphics mode selection

The word **gr** selects the graphics mode. This word makes visible a canvas-type graphic space:

```
web gr
```

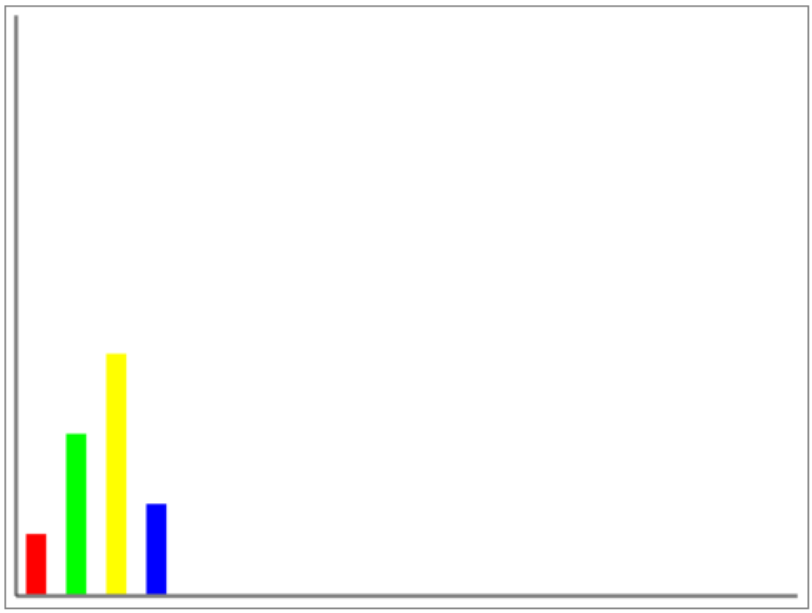
To return to text mode, type **text** .

Here is an example of a graphic:

```
web
: grTest ( -- )
    gr 400 300 window
    $000000 color!
    5 295 395 295 line
    5 295   5   5 line
    $ff0000 color! 10 294 10  -30 box
    $00ff00 color! 30 294 10  -80 box
    $ffff00 color! 50 294 10 -120 box
    $0000ff color! 70 294 10  -45 box
    key drop
    text
  ;
```

Running **grTest** displays this graph:

Try eFORTH



uEforth v7.0.7.9 - rev fb3db70da6d111b1fdf0

Creating FORTH Definitions Using JavaScript

The combination of JavaScript and FORTH

eFORTH web works in all web browsers, on all Windows, Linux, MacOS, and Android systems... How is this possible?

The secret to this extraordinary versatility is simple: a successful combination of JavaScript and FORTH.

Most of the FORTH language primitives are written in a pseudo-assembler or in Forth itself. We will see how to expand the web vocabulary by defining the word `ellipse`:

```
web definitions

\ draw ellipse in graphic mode
JSWORD: ellipse { x y rx ry angle div  }
  context.ctx.ellipse(x, y, rx, ry, Math.PI * 2 * angle / div, 0, 2 *
Math.PI);
~
forth definitions
```

Here, the word `ellipse` is defined after the creation word `JSWORD:`. The end-of-definition delimiter is marked by the `~` character. Let's examine the structure of our `ellipse` definition in detail.

Structure of a word definition with `JSWORD:`

Let's analyze the first line of the `ellipse` definition:

```
web definitions
JSWORD: ellipse { x y rx ry angle div  }
  context.ctx.ellipse(x, y, rx, ry, Math.PI * 2 * angle / div, 0, 2 *
Math.PI);
~
```

- The word `JSWORD:` marks the beginning of a definition using JavaScript code. This definition word is immediately followed by the word being defined, in this case, `ellipse`.
- The word `ellipse` is the word being defined. It is followed by parameter passing using local variables.
- The parameters are passed using `{ x y rx ry angle div }`.

The following line contains the JavaScript code:

```
JSWORD: ellipse { x y rx ry angle div  }
```

```
  context.ctx.ellipse(x, y, rx, ry, Math.PI * 2 * angle / div, 0, 2 *
Math.PI);
~
```

JavaScript code can also be written over multiple lines:

```
web definitions
JSWORD: time@ { -- h m s }
    let date = new Date();
    var hh = date.getHours();
    var mm = date.getMinutes();
    var ss = date.getSeconds();
    return [hh, mm, ss];
~

JSWORD: date@ { -- y m d }
    let date = new Date();
    var yy = date.getFullYear();
    var mm = date.getMonth()+1;
    var dd = date.getUTCDate();
    return [yy, mm, dd];
~
forth definitions
```

Now, let's take a detailed look at parameter passing.

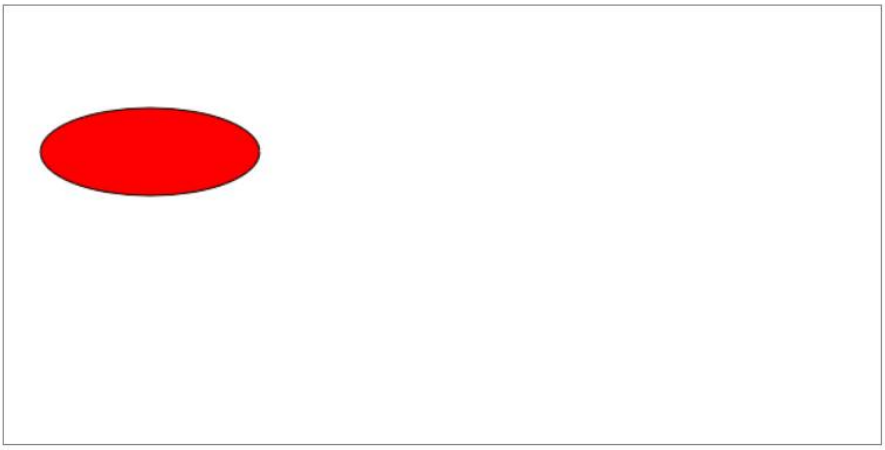
Parameter passing petween eFORTH and JavaScript

Parameter passing uses local variables. These variables are defined between `{` and `}`. Let's revisit the `ellipse` word definition:

```
JSWORD: ellipse { x y rx ry angle div  }
  context.ctx.ellipse(x, y, rx, ry, Math.PI * 2 * angle / div, 0, 2 *
Math.PI);
~
```

Here, we pass five parameters. These parameters are then used in a FORTH language definition:

```
web gr
600 300 window
$ff0000 color!
100 100 75 30 0 360 ellipse  fill
0 color!  stroke
```



There are four cases of parameter passing to a word defined by **JSWORD:**:

1. Calling a JavaScript function without passing any parameters. The braces are empty:

```
JSWORD: beginPath { }
  context.ctx.beginPath();
~
```

2. Calling a JavaScript function with input parameters only:

```
JSWORD: lineTo { x y }
  context.ctx.lineTo(x, y);
~
```

3. Calling a JavaScript function that only retrieves output parameters:

```
JSWORD: viewport@ { -- w h }
  return [context.width, context.height];
~
```

4. Passing both input and output parameters:

```
JSWORD: web-type-raw { a n -- yld }
  if (globalObj.write) {
    var text = GetString(a, n);
    write(text);
    return 0;
  } else {
    var newline = false;
    for (var i = 0; i < n; ++i) {
      var ch = u8[a + i];
      if (ch == 10) { newline = true; }
      context.Emit(ch);
```

```
    }
    if (newline) {
      context.Update();
    }
    return newline ? -1 : 0;
  }
~
```

In this last case, input and output parameters are separated by `--`. Normally, in a standard eFORTH definition, these characters and what follows are ignored. However, for `JSWORD:`, parameter passing must strictly follow one of the four forms listed above.

For output parameters, it is not necessary to use JavaScript variables. Let's simplify the `time@` function:

```
web definitions
JSWORD: time@ { -- h m s }
    let date = new Date();
    var hh = date.getHours();
    var mm = date.getMinutes();
    var ss = date.getSeconds();
    return [hh, mm, ss];
~
forth definitions
```

Extending eFORTH's Web Vocabulary

The web vocabulary of eFORTH is quite limited. You can extend it by retrieving the contents of the file `additionalDefs.fs` from the `eForthWEB-book.zip` file.

Install this file in a directory of your choice. We prefer the directory name `fs` (for "Forth Script"). To call the contents of this file `additionalDefs.fs` from your web application:

```
<div id="ueforth"></div>
<script src="../js/ueforth.js"></script>
<script type="text/forth" src="fs/additionnalDefs.fs"></script>
<script type="text/forth">
\ here additionnal Forth Code
</script>
```

Version v 7.0.7.15

FORTH

-	-rot	,	;	:	:noname	!	?
?do	?dup	.	."	.s	'	(local)	[
[']	[char]	[ELSE]	[IF]	[THEN]	]	{	}transfer
@	*	*/	*/MOD	/	/mod	#	#!
#>	#fs	#s	#tib	+	+!	+loop	+to
<	<#	<=	<>	=	>	>=	>BODY
>flags	>flags&	>in	>link	>link&	>name	>name-length	
>params	>R	>size	0<	0<>	0=	1-	1/F
1+	2!	2@	2*	2/	4*	2drop	2dup
4/	abort	abort"	abs	accept	afliteral	aft	again
ahead	align	aligned	allot	also	AND	ansi	ARSHIFT
asm	assert	at-xy	base	begin	bg	binary	bl
blank	bye	c,	C!	C@	CALL	CASE	catch
CELL	cell/	cell+	cells	char	cmove	cmove>	colors
CONSTANT		context	cr	CREATE	current	decimal	defer
DEFINED?		definitions	depth	do	DOES>	DROP	dump
DUP	echo	else	emit	ENDCASE	ENDOF	erase	evaluate
EXECUTE	EXIT	exit	extract	F-	f.	f.s	F*
F**	F/	F+	F<	F<=	F<>	F=	F>
F>=	F>S	F0<	F0=	FABS	FATAN2	fconstant	FCOS
fdepth	FDROP	FDUP	FEXP	fg	fill	fill32	FIND
fliteral		FLN	FLOOR	FMAX	FMIN	FNEGATE	FNIP
for	forget	FORTH	forth-builtins		FOVER	FP!	FP@
fp0	FROT	FSIN	FSINCOS	FSQRT	FSWAP	fvariable	handler
here	hex	hld	hold	I	if	IMMEDIATE	internals
invert	is	J	K	key	key?	L!	latestxt
leave	literal	loop	LSHIFT	max	min	mod	ms
ms-ticks		n.	negate	nest-depth	next	nip	nl
normal	octal	OF	ok	only	OR	order	OVER
pad	page	PARSE	pause	PI	postpone	precision	previous
prompt	quit	r"	R@	R>	r|	r~	rdrop
recurse	refill	remaining	repeat	rot	RP!	RP@	rp0
RSHIFT	s"	S>F	s>z	sealed	see	set-precision	
set-title		sf,	SF!	SF@	SFLOAT	SFLOAT+	SFLOATS
sign	SL@	SP!	SP@	sp0	space	spaces	start-task
startswith?		state	str	str=	structures	SW@	SWAP
task	tasks	terminate	then	throw	tib	to	transfer
transfer{		type	u.	U/MOD	UL@	UNLOOP	until
used	UW@	value	VARIABLE	vlist	vocabulary	W!	web
while	words	XOR	z"	z>s			

ansi

terminal-restore terminal-save show hide scroll-up scroll-down clear-to-eol
bel esc

asm

code, code4, code3, code2, code1, callot chere code-at code-start

internalized

flags'or! LEAVE LOOP +LOOP ?DO DO NEXT FOR AFT REPEAT WHILE ELSE IF THEN AHEAD UNTIL AGAIN BEGIN cleave

internals

DOFLIT S>FLOAT? 'heap 'context 'latestxt 'notfound 'heap-start 'heap-size 'stack-cells 'boot 'boot-size 'tib 'argc 'argv 'runner 'throw-handler NOP BRANCH 0BRANCH DONEXT DOLIT DOSET DOCOL DOCON DOVAR DOCREATE DODOES ALITERAL LONG-SIZE S>NUMBER? 'SYS YIELD EVALUATE1 'builtins internals-builtins cases (+to) (to) --? }? ?room scope-create do-local scope-clear scope-exit local-op scope-depth local+! local! local@ <>locals locals-here locals-area locals-gap locals-capacity ?ins. ins. vins. onlines line-pos line-width size-all size-vocabulary
vocs. voc. voclist voclist-from see-all >vocnext see-vocabulary nonvoc? see-xt ?see-flags see-loop see-one indent+! icr see. indent mem= ARGS_MARK -TAB +TAB NONAMED BUILTIN_FORK SMUDGE IMMEDIATE_MARK dump-line ca@ cell-shift cell-base cell-mask CALLCODE #f+s internalized BUILTIN_MARK zplace $place free. boot-prompt raw-ok [SKIP]' [SKIP] ?stack sp-limit input-limit tib-setup raw.s $@ digit parse-quote leaving, leaving)leaving leaving(value-bind evaluate&fill evaluate-buffer arrow ?arrow. ?echo input-buffer immediate? eat-till-cr wascr *emit *key notfound last-vocabulary voc-stack-end xt-transfer xt-hide xt-find& scope

structures

field struct-align align-by last-struct struct long ptr i64 i32 i16 i8 typer last-align

tasks

.tasks main-task task-list

web

yielding-task yielding import rm ls download cat include-file upload upload-file session? web-key? web-key web-type scripts scripts# random button mouse textWidth fillText font text-size! log raw-http-upload http-download raw-download upload-success? upload-done? upload-start ms-ticks silence tone importScripts release keyCount getKey clearItems removeItem getItem setItem smooth gpop gpush rotate scale translate show-text keys-height mobile textRatios viewport@ window line fill stroke lineTo moveTo beginPath box lineWidth color! text gr grmode shouldEcho? web-terminate web-key?-raw web-key-raw web-type-raw jseval JSWORD: jsslot jseval!

Lexical index

ansi.....10
asm.....11
FORTH.....10
internalized.....11
internals.....11
JSWORD:.....6
structures.....11
tasks.....11
web.....11