DMV005 DMetashell: A Full Stack Shell with One Liner Metaprogramming Script at 2.5mb Uncompressed

In this article, we demonstrate how DMetashell may produce a 3x3 html table (red arrow in figure 1) with one line of metaprogramming script in Phoscript, derived from FORTH programming language:

f(': B_F htab33 hje ; AJE')

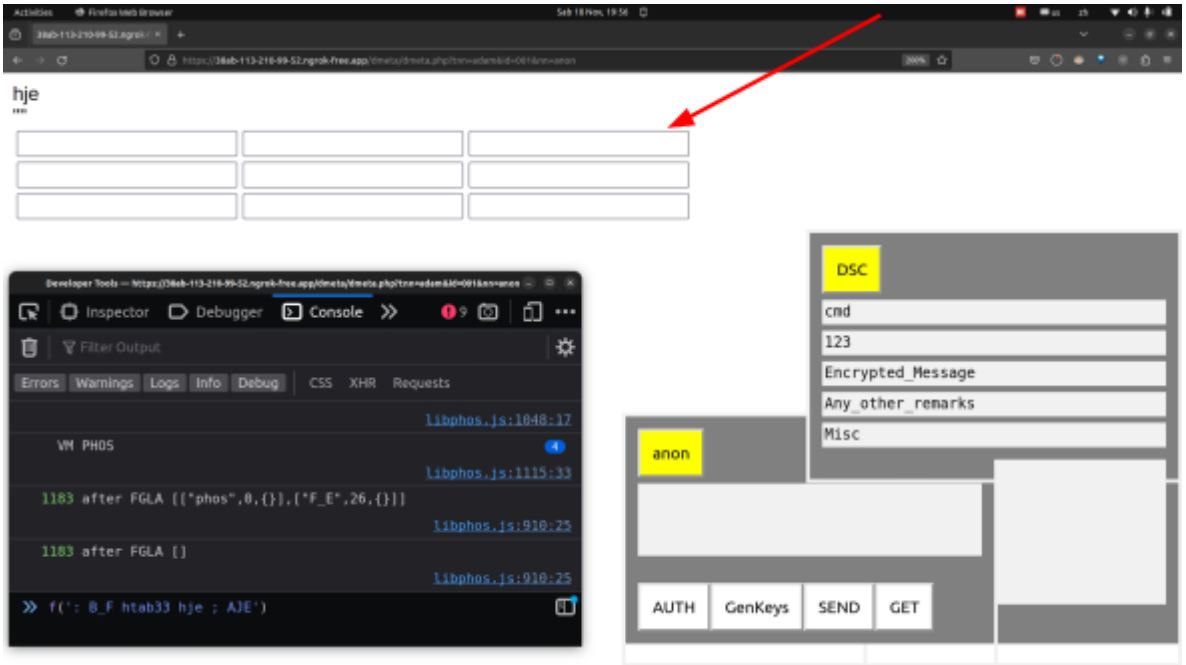

Figure 1

In the past decade or so, development of web and mobile device programming frameworks have largely moved in a similar direction, loosely based on Model-View-Controller framework, with layers upon layers of increasingly complex libraries and exotic programming languages, resulting in what can be euphemistically called “gigabyte hello world” apps (LOL, Monty Python style!)

We took a drastically opposite direction, based on a relatively obscure FORTH programming language, to derive a single-function virtual machine and its metaprogramming script Phoscript (Greek for “light”, a pun for English “lack of weight”), (theoretically possible) portable to any know programming language as “host”, resulting in what is described in the title:

> DMetashell: A Full Stack Shell with One Liner Metaprogramming Script at 2.5mb Uncompressed

Of course, 2.5mb of JavaScript and PHP source for DMetashell is in its unoptimised state. Readers are welcome to take up the challenge to optimise them.

What we hope to achieve are illustrated in figures 2 and 3:

A. 1 billion HCCP (homoiconic command-comment pair, a fundamental measure of code) in DMetashell by 2030.
B. At least 0.1% of MAGA (Microsoft, Meta, Amazon, Google, Apple) revenues by 2030.

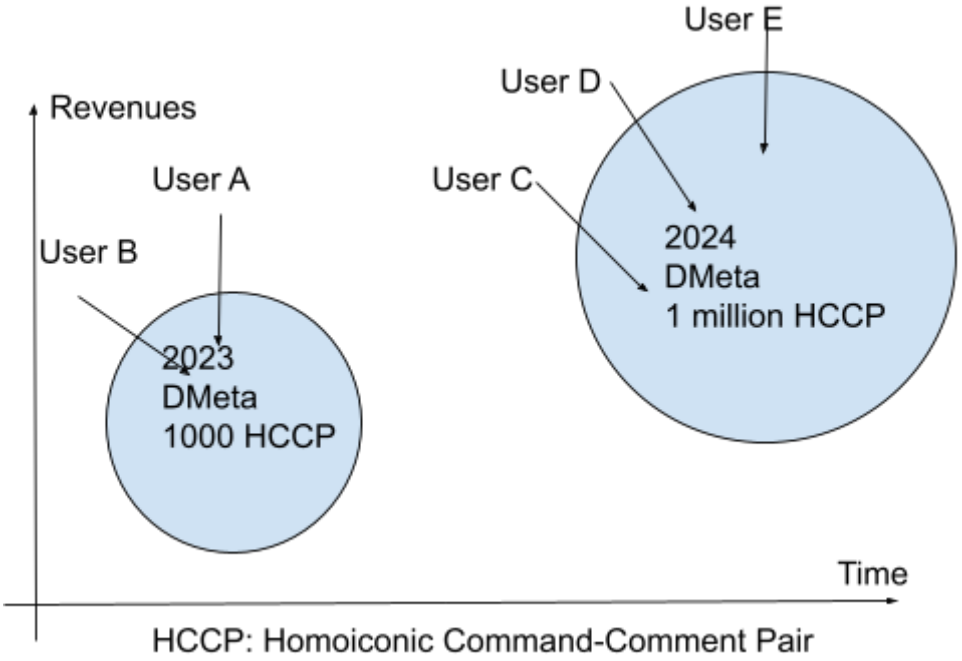

Figure 2

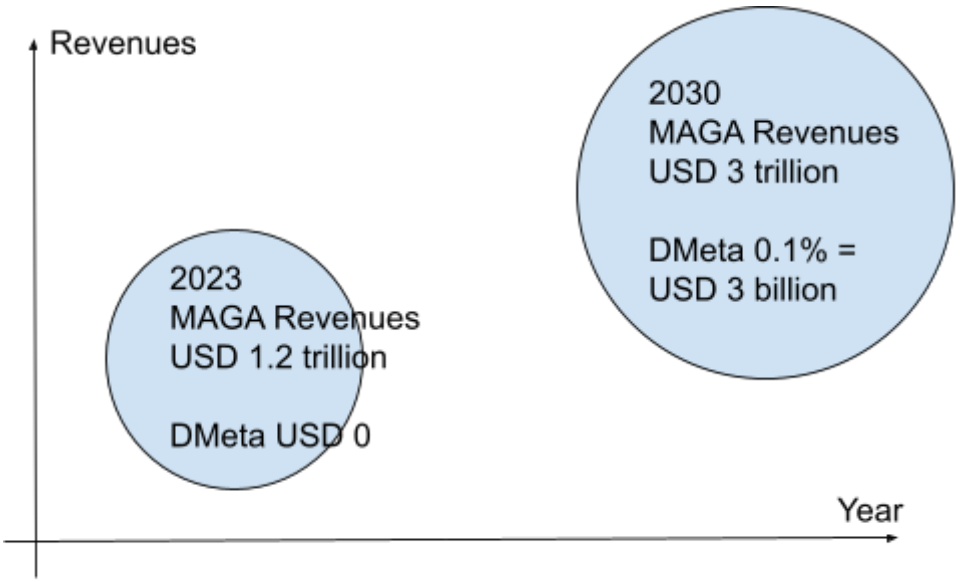

Figure 3

1. The command concerned in this article is:

   f(': B_F htab33 hje ; AJE')

The definition of *htab33* is given in the file Graph/hg/xhash (figures 4 and 5).

```
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$ sed 's/,/,\n/g' Graph/hg/xhash
2023-11-06T17:23:53.592+0000 {"comments":"Graph\\/hg\\/A4FfkCU_-A==",
"comments_je":"Graph/hg/AlV20bbKog==",
"htab33":"Graph/hg/Et3ygPECUQ==",
"hgst":"Graph/hg/CpoFl52tIg==",
"hgsd":"Graph/hg/HpsPgl4WNQ==}
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$
```

Figure 4

```
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$ cat Graph/hg/Et3ygPECUQ==
2023-11-14T12:54:32.324+0000 f(': cell a b input: th tn: ; : row cell cell cell 3 ms: tr tn: ; body getn:
 0 i: div ce: row row row 3 ms: table tn: ih: ac:')
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$
```

Figure 5

   f(': cell a b input: th tn: ; : row cell cell cell 3 ms: tr tn: ; body getn: 0 i: div ce: row row row 3 ms: table tn: ih: ac:')

f() is a JavaScript function to execute the argument in Phoscript interpreter, which is essentially a stack machine, running on reverse polish notation in the convention of FORTH programming language. The string can be broken down as follow: (\ marks the start of comments)

```
: cell a b input: th tn: ;          \ colon definition (function) for cell
a b input:        \ create input element id="b" name="a"
th tn:            \ create tag <th>

: row cell cell cell 3 ms: tr tn: ; \ colon definition for row
cell cell cell    \ execute cell 3 times
3 ms:             \ merge 3 elements
tr tn:            \ make html tag <tr>

body getn:        \ getElementsByTagName( 'body' )
0 i:              \ extract the first element from the array
div ce:           \ create element of type div
row row row       \ execute row 3 times
3 ms:             \ merge 3 elements
table tn:         \ make html tag <table>
ih:               \ set innerhtml
ac:               \ appendChild()
```

Figure 6 shows fgl_input() (line 429) and fgl_ac() (line 432) which defines input: and ac: in Phoscript respectively. Readers may look up source code of DMetashell for details of remaining commands.

```
422         S.push(b[a]);
423     }
424     function fgl_ix() {
425         var a = S.pop();
426         var b = S[S.length - 1];
427         S.push(b[a]);
428     }
429     function fgl_input() {
430         S.push('<input type="text" id="' + S.pop() + '" name="' + S.pop() + '">');
431     }
432     function fgl_ac() {
433         var child = S.pop();
434         var parent = S.pop();
435         parent.appendChild(child);
436         S.push(parent);
437     }
438     function count(a) {
439         return a.length;
440     }
441     function fgl_explode() {
442         S.push(explode(S.pop(), S.pop()));
443     }
444     function fgl_now() {
445         var d = new Date();
446         S.push(d.toISOString());
447     }
448     function fgl_colon() {
449         S.push(':');
450     }
451     function fgl_timeout() {
```

Figure 6

The previous section describes Phoscript mappings to JavaScript in the front end.

In the next section, we show how Phoscript maps to PHP functions in the back end.

Reminding ourselves on the command concerned:

f(': B_F htab33 hje ; AJE')

htab33 is the input parameter for *hje*, where the operand precedes the operator (function), in reverse polish notation.

hje is defined as shown in figure 7, which essentially reads the contents of the file as shown in figure 5. Figure 8 shows execution output in the back end.

```
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$ cat o_cdw.json | sed 's/],/],\n/g' | grep "cmdh\|
hje"
"cmdh":["Graph\/hg\/xhash","fgc:","dup:","{\"","swap:","pos:","substr:","jd:","1","pick:","i:","sslh:",";
"],
"xhje":["fgc:","dup:","php","swap:","pos:","substr:","exec:","array:","ON","ECHO","bv:","map:","nop:","je
:",";"],
"xh":["cmdh","xhash",";"],
"hje":["cmdh","fgc:","dup:","29","substr:","je:","ON","ECHO","bv:","enl:",";"],
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$
```

Figure 7

```
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$ htab33 hje
It's my handle!
Arguments: htab33 hje
"f(': cell a b input: th tn: ; : row cell cell cell 3 ms: tr tn: ; body getn: 0 i: div ce: row row row 3
ms: table tn: ih: ac:')\n"
hongwu@hongwu-Latitude-5480:~/devel/2023/02/dmeta/auth$
```

Figure 8

Finally, the definition of *AJE* (Execute (eval()) AJAX results) is given in lines 86 and 88 in figure 9.

```
65 // need to have function to load skp from S0.KS.pair to stack: l_sign_kp:
66 // then make function to import skp from json, then save to S0.KS.pair, load to stack when needed
67 f(': l_rsa_sign privateKey ix: dsc_sign geid: value: awa: sign: l_do_scj ;')
68 f(': l_dsc a_uuid: 3 pick: dup: awa: l_do_sign ss: ;') // change this to test Ciphertext change
69
70 // Get Hash of PBK
71 // M.F(': B_AUTH1 SESSION: AUTH i: PBK i: o_hash.json l_cdwjs h_b64 s: ; : A_AUTH1 B: B_AUTH1 F_O ; A_AUTH1');
72 hpbk = ': B_AUTH1 SESSION: AUTH i: PBK i: o_hash.json l_cdwjs h_b64 s: ; : A_AUTH1 B: B_AUTH1 F_O ; A_AUTH1'
73
74 // M.F(': B_AUTH1 HTTP SESSION: AUTH i: PBK i: o_hash.json l_cdwjs h_b64 dup: last rg: newest 2 4 sspl: s: ; :
   A_AUTH1 B: B_AUTH1 F_O ; A_AUTH1');
75 newest_msg = ': B_AUTH1 HTTP SESSION: AUTH i: PBK i: o_hash.json l_cdwjs h_b64 dup: last rg: newest 2 4 sspl: s: ; :
   A_AUTH1 B: B_AUTH1 F_O ; A_AUTH1';
76
77 // f('S0.K ev: je: blob: keypair-chris-b.json download:')
78 savekp = 'S0.K ev: je: blob: keypair-chris-b.json download:'
79
80 // libphos.js:M.F(': AJAXP B: B_F F_P ;');
81 // M.F(': AJE B: B_F F_E ;');
82 // M.F(': F_E dup: ;');
83 // M.F(':    ;') bug missing ; in output
84 // M.F(': F_E dup: jd: ev: ;');
85
86 f(': AJE B: B_F F_E ;');
87 // f(': F_E dup: ;');
88 f(': F_E dup: jd: ev: ;');
89
90 // New browser 20230616
91 // insert dummy item to S so that S[2] is now nickname dialogue box
92 S.splice(1,0,S[1])
```

Figure 9

Figure 10 shows a snapshot of “grep -r 2023 Graph/hg” results, which form the hash-graph database by using hash strings as the filenames for each line of the Homoiconic Command-Comment Pair (HCCP).

```
Activities   Terminal                                   Sab 18 Nov, 20:55
hongwu@hongwu-Latitude-5480: ~/devel/2023/02/dmeta/auth
Graph/hg/ExO6VRqEnA==:2023-11-06T17:01:54.771+0000 table of command to hash for execution
Graph/hg/DNgeqAbBUA==:2023-11-06T17:23:53.592+0000 {"comments":"Graph\\/hg\\/A4FfkCU_-A==","comments_je":
"Graph/hg/AlV20bbKog==","htab33":"Graph/hg/Et3ygPECUQ==","hgst":"Graph/hg/CpoFl52tIg==","hgsd":"Graph/hg/
HpsPgl4WNQ==="}
Graph/hg/Ak-PAin8Rg==:2023-11-06T17:24:45.635+0000 JSON for command to hash for execution
Graph/hg/ASQJ7ld-7g==:2023-11-06T17:44:47.047+0000  2474  php phos.php comments cmdh xhash
Graph/hg/GS9jap3Y5Q==:2023-11-06T17:45:45.504+0000 Command to hash, then execute
Graph/hg/HPokVDd4RA==:2023-11-06T17:49:03.411+0000  2482  comments xh
Graph/hg/FfQSy99GlA==:2023-11-06T17:49:52.438+0000 Command to hash then execute, macro xh
Graph/hg/HjY7NdMKyw==:2023-11-06T18:43:02.475+0000  2472  php phos.php comments Graph/hg/xhash fgc: dup:
'{"' swap: pos: substr: jd: 1 pick: i: sslh: 3 pick: je: s:
Graph/hg/HvWOz08Tmg==:2023-11-06T18:44:01.041+0000 Pick 0 element of $S, je: for easy editing in o_cdw.js
on
Graph/hg/AlV20bbKog==:2023-11-08T00:48:01.099+0000  2549  php phos.php hgnc: dup: /php.phos/ g: dup: ak:
array: ON ECHO bv: : f1 dup: 5 pick: swap: 1 + i: array: swap: ap: swap: 5 pick: swap: i: ap: \; map: f1
je: enl:
Graph/hg/GsLOo1qtEw==:2023-11-08T00:48:22.594+0000 comments no color je:
Graph/hg/Dbm5UuwSmg==:2023-11-14T12:54:12.834+0000 command_not_found_handle() {     echo "It's my handle!
";     echo "Arguments: $@"; php phos.php $@ }
Graph/hg/Et3ygPECUQ==:2023-11-14T12:54:32.324+0000 f(': cell a b input: th tn: ; : row cell cell cell 3 m
s: tr tn: ; body getn: 0 i: div ce: row row row 3 ms: table tn: ih: ac:')
Graph/hg/CoHmymq50Q==:2023-11-14T12:54:56.943+0000 3 x 3 rows of input cells in html
Graph/hg/GFprREH4TQ==:2023-11-14T17:44:03.933+0000 f(': B_F '+  JSON.stringify(M.S[0].$CDW.l_1)  +' jd: l
_1 acdw: htab33 hgje ON ECHO bv: enl: ; AJE')
Graph/hg/C67mHyGG7g==:2023-11-14T17:44:59.852+0000 Back end sends JSON to front end, AJE push JSON to sta
ck without display.
:
```

Figure 10

Finally, we copy the final results in figure 1 below, which demonstrate how DMetashell may produce a 3x3 html table (red arrow in figure 1) with one line of metaprogramming script in Phoscript, derived from FORTH programming language:

f(': B_F htab33 hje ; AJE')

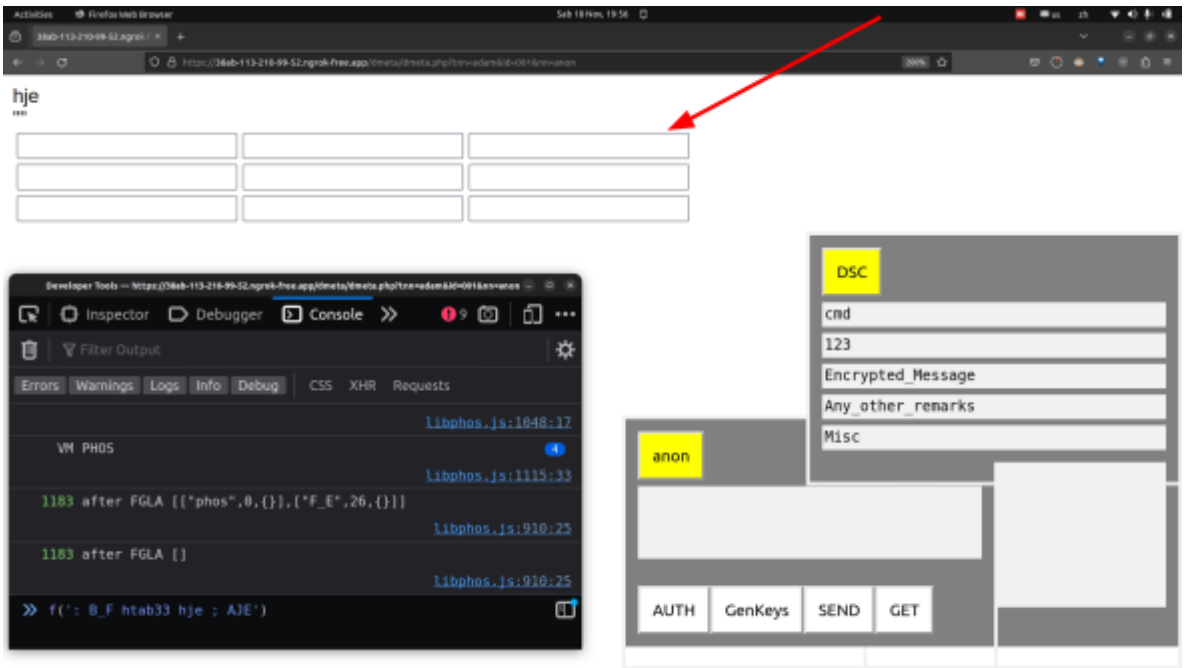

Figure 1

References

DMV004 An Incremental Homoiconic Documentation System for DMetashell
https://godmeta.github.io/doc/DMV004_Homoiconic_Documentation_for_DMetashell.pdf

DMV003 DMetashell & Phoscript Stack Machine
https://godmeta.github.io/doc/DMV003_DMetashell%20_Phoscript_Stack_Machine.pdf

DMV002 Build YOUR OWN Decentralised Metaverse with DMetashell
https://godmeta.github.io/doc/DMV002_Build_Decentralised_Metaverse_DMetashell.pdf

Appendix A: DMeta Hash Contract – Creating Contracts using Hash Values
https://godmeta.github.io/doc/DMeta_Hash_Contract_Transaction.pdf