RFD: Place new words from proposals into an individual namespace.

Problem

New ideas propose words that may be already in use in some Forth's. Same with concurrent proposals for an existing problem. They may have the same words with different meanings.

Solution

Make use of named wordlists to create distinguished namespaces. Traditionally with vocabularies, more modern with wordlists.

Proposal

Every proposal creates it's own wordlist and defines a named wordlist identifier. The name of this wordlist is the name of the proposal. The proponent makes sure that there is no other proposal of that name already. All words from the proposal are in that wordlist. This wordlist can be added to the text interpreter with e.g. the search order or with other techniques like a dot-parser. That way even replacements or alternative implementations of existing standard words can be dealt with.

The name "FORTH" is reserved for the standard words and must not be used by proposals.

If a proposal is considered for inclusion into the standard the names of the words are checked and adapted accordingly by the editor. (S)he gets suggestions from the author that shall be used.

Experience

All systems of relevance (that is: used by other people than their respective authors) have the search order wordset included. Those that don't often provide an optional package for it (e.g. noforth).

All existing RFD's operate in the FORTH namespace. That already did create collisions with other non-standard system specific words. Rumors are that this collisions did slow down the acceptance of otherwise good ideas.

Typical Usage

A wordlist can be either added to the search order stack to make the words available to the text interpreter. Alternatively a "dot-parser" can be used to avoid search order problems.

The examples use both wordlists or vocabularies. A clean wordlist only implementation that avoids vocabularies (they are not part of the standard) is available too.

```
vocabulary foo
get-current
also foo definitions
\ define the words
: bar 1 ;
variable baz
....
set-current \ restore the previous current wordlist

T{ bar -> 1 }T
T{ 42 baz ! -> }T
T{ baz @ -> 42 }T
```

The dot-parser approach requires more assistance from the command interpreter. The example below uses the dot-parser recognizer found in e.g. gforth's rec-scope.fs

```
vocabulary foo
get-current
also foo definitions

: bar 1 ;
variable baz
....

set-current previous

T{ foo:bar -> 1 }T
T{ 42 foo:baz ! -> }T
T{ foo:baz @ -> 42 }T
```

A further syntactic simplification is the IN word discussed at the Euroforth 2016. With that all of the search order management is hidden and the example from above turns into

```
vocabulary foo

IN foo : bar 1 ;
IN foo variable bar

T{ foo:bar -> 1  }T
T{ 42 foo:baz ! -> }T
T{ foo:baz @ -> 42 }T
```