

Ownership

Vlastnění objektu

Systém vlastnických práv objektů je rozšíření podpory událostí objektů. Nyní může objekt vlastnit série, na něž odkazuje, dokonce i série vnořené. Mění-li se vlastněná série, je uvědoměn vlastníci objekt a je případně volána jeho funkce `on-deep-change*`, umožňující jeho příslušnou reakci.

Prototypem pro `on-deep-change*` je:

```
on-deep-change*: func [owner word target action new index part][...]
```

Argumenty jsou:

- owner: objekt přijímající událost (object!)
- word: slovo objektu, odkazující na měněné série (řady) nebo vnořené série (word!)
- target: měněné série (any-series!)
- action: název aplikované akce (word!)
- new: nová hodnota, přidána k sérii (any-type!)
- index: pozice, na níž je série modifikována (integer!)
- part: počet měněných elementů v sérii (integer!)

Názvem akce může být: `random`, `clear`, `cleared`, `poke`, `remove`, `removed`, `reverse`, `sort`, `insert`, `take`, `taken`, `swap`, `trim`. U akcí, které "rozbíjí" hodnoty, jsou generovány dvě události, jedna před a druhá po "destrukci" (odtud přítomnost slov `cleared`, `removed`, `taken`).

Jestliže modifikace ovlivňuje několik nesousedících nebo všechny elementy, je `index` nastaven na hodnotu -1. Jestliže modifikace ovlivňuje neurčený počet elementů, je `'part'` nastaven na hodnotu -1.

Ownership je nastaven automaticky při vytvoření objektu, je-li definována funkce `on-deep-change*`, načež jsou všechny odkazované série (včetně vnořených) vlastněny vytvořeným objektem. Byla zavedena také akce `modify`, umožňující ustavení nebo zrušení vlastnictví po vytvoření objektu.

As for `on-change`, `on-deep-change*` is kept hidden when using `mold` on an object. It is only revealed by `mold/all`.

Při použití objektu ve funkci `mold` jsou funkce `on-change`, `on-deep-change*` skryté (hidden).

Zde je jednoduchý příklad použití vlastnictví objektu. Níže uvedený kód vytvoří řadu objektů, obsahujících prázdný seznam. K němu lze připojit pouze celé číslo. Připojí-li se hodnota jiného typu, zobrazí se sdělení (message) a neplatný element je ze seznamu odebrán. Seznam navíc zůstává vždy uspořádaný (sorted) po připojení (insert, poke) nové hodnoty:

```
numbers: object [  
  list: []  
  
  on-deep-change*: function [owner word target action new index part][
```

```

        if all [word = 'list find [poke insert] action][
            forall target [
                unless integer? target/1 [
                    print ["Error: Item" mold target/1 "is invalid!"]
                    remove target
                    target: back target
                ]
            ]
            sort list
        ]
    ]
]

red>> append numbers/list 3
== [3]
red>> insert numbers/list 7
== [3 7]
red>> append numbers/list 1
== [1 3 7]
red>> insert next numbers/list 8
== [1 3 7 8]
red>> append numbers/list 4
== [1 3 4 7 8]
red>> append numbers/list "hello"
Error: Item "hello" is invalid!
== [1 3 4 7 8]
red>> numbers
== make object! [
    list: [1 3 4 7 8]
]

```

Vlastnictví objektu se hojně používá v Red/View za účelem vytvoření vazby mezi objekty **face** a widgety na obrazovce, jakož i pro zajištění automatické synchronizace bez potřeby volat **show**.

Práce v této oblasti není ukončená. Očekává se, že bude zavedeno více událostí, s možností objektů vlastnit více datových typů. *More documentation will be provided once the work on that is finished. In the future, its use will be extending to other frameworks and interfaces. Such "reactive objects" will be called "live objects" in Red's jargon.*