

Re: lambda... again

Source: <http://coding.derkeiler.com/Archive/Tcl/comp.lang.tcl/2007-10/msg00900.html>

- *From:* "slebetman@xxxxxxxx" <slebetman@xxxxxxxx>
 - *Date:* Tue, 23 Oct 2007 01:00:37 -0700
-

On Oct 23, 3:32 pm, suchenwi <richard.suchenwirth-bauersa...@xxxxxxxxxxxx> wrote:

On 23 Okt., 03:22, "slebet...@xxxxxxxx" <slebet...@xxxxxxxx> wrote:

The disadvantage is that calling a lambda is different from calling a regular proc (you need to prepend apply to it). Now... if someone feels like modifying [unknown] so that lambdas can be called like a proc.....

Oh, that's easily done:

```
% proc know what {
if ![info complete $what] {error "incomplete command(s) $what"}
proc unknown args $what\n[info body unknown]
}
% know {
if {[llength [lindex $args 0]]==2} {return [eval [linsert $args 0
apply]]}
}
% set try {{a b} {expr {$a * $b}}}
{a b} {expr {$a * $b}}
% $try 7 6
42
```

It just conflicts with spaces in command names, so don't do both... :^)

HA! Now that we have this we can TRULY treat everything as a string. Proc is no longer necessary. Just store everything in variables!

OTOH since variables are locally scoped writing tcl code in pure lambdas is going to be annoying:

```
set something {{} {
global do_something print ;# must "import" all lambdas before
using :-(
```

Re: lambda... again

```
$do_something here  
$print Done!  
}}
```

```
set do_something {{x} {  
global print ;# typing global all the time quickly gets annoying  
$print "doing something: $x"  
}}
```

```
set print {x {puts "$x\n"}}
```

```
$something
```

.