REPORT and STOP BLOCK primitives in Snap!

Jens Mönig, Feb 24, 2012

This document explains the intention behind the current implementation of the REPORT and STOP BLOCK primitives today's pre-alpha („nasciturus") version of Snap! Please refer to these rules when testing your own projects or custom blocks - or Snap! - for correctness.

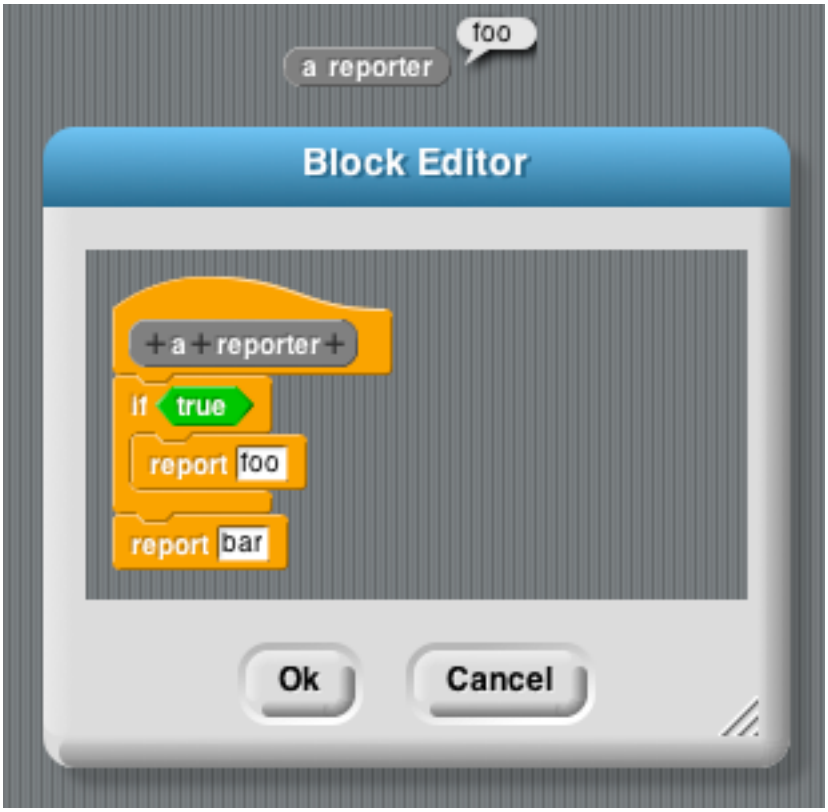
The Rule

> When a REPORT block is evaluated it returns its value to the outside of the nearest enclosing custom block, or - in the absence of such - to the outside of the nearest enclosing lambda expression - or - in the absence of such - it stops the current script.

Note that each part of this rule takes precedence over the following ones.

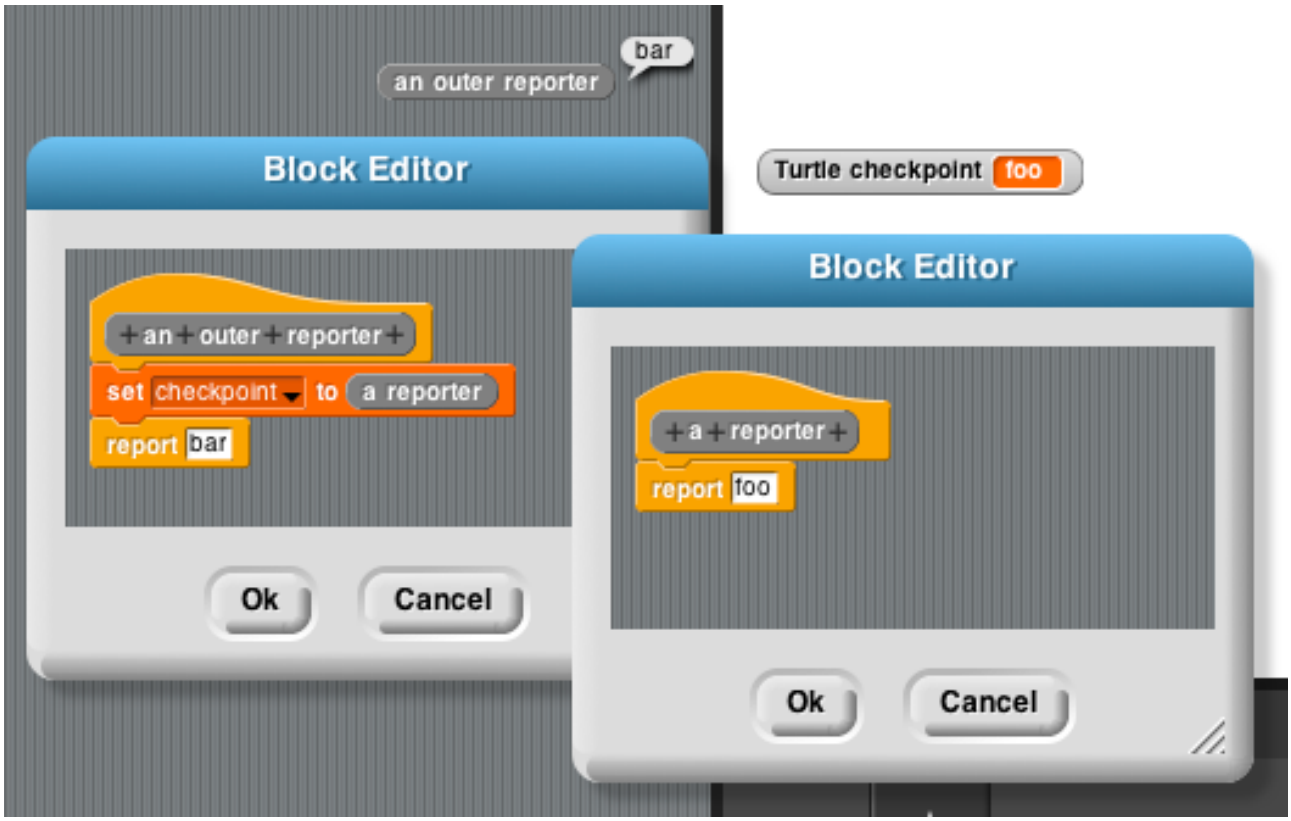
REPORT Examples

„When a REPORT block is evaluated it returns its value to the outside of the nearest enclosing custom block":



Since the IF condition is true evaluating REPORT (foo) stops the reporter and returns foo to its continuation („outside"). REPORT (bar) doesn't get evaluated.

This lets custom reporters be nested arbitrarily inside of each other‘s definition bodies:

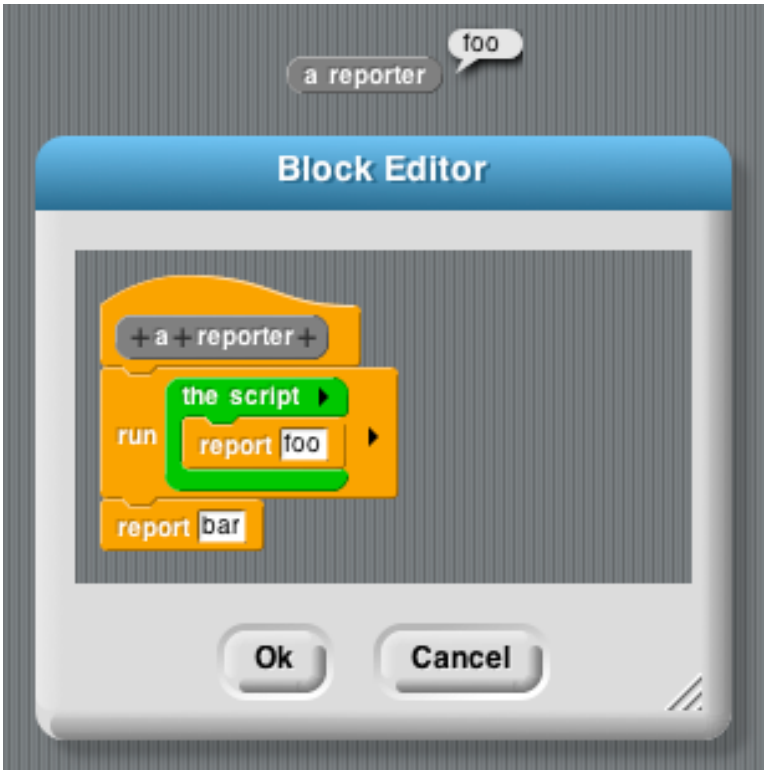


in this example, REPORT (foo) only returns to the outside of the A REPORTER block, therefore the enclosing AN OUTER REPORTER‘s definition body continues to be evaluated and eventually reporters „bar“.

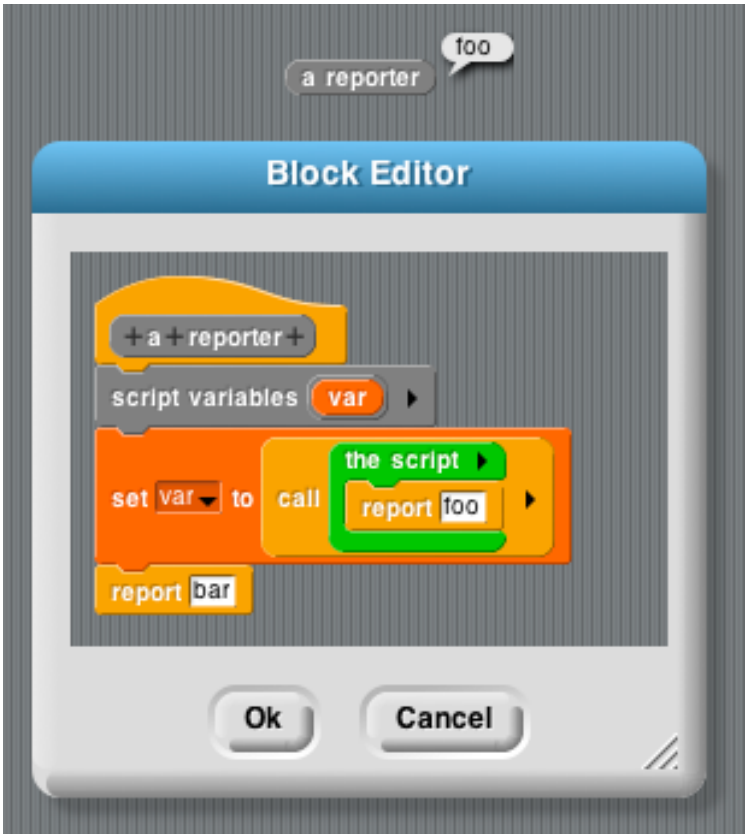
„... in the absence of such - to the outside of the nearest enclosing lambda expression“:

Note that in this example the blocks are not located inside a Block Editor but within the Turtle‘s general scripting area. Therefore evaluating REPORT (foo) only returns to the outside of the THE SCRIPT block and both checkpoints get passed.

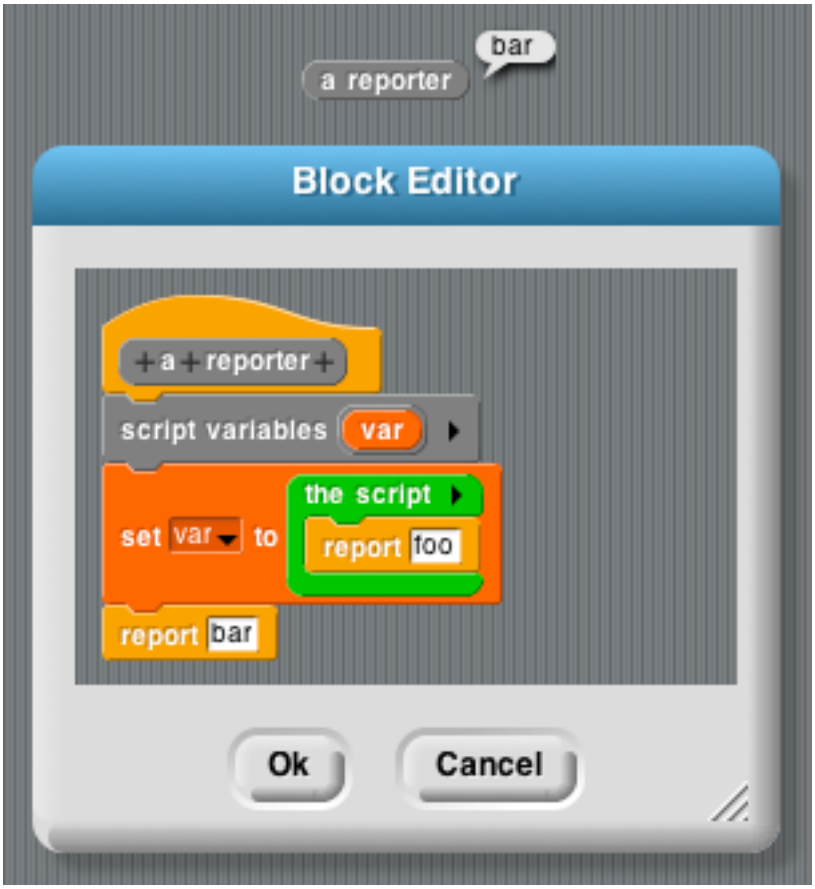
However, since the custom block rule takes precedence over the lambda rule, evaluating REPORT inside a custom block's body always stops it:



Even when REPORT is evaluated inside an assignment statement (inside a block's definition):



Bear in mind, though, that the rule only applies to REPORT blocks which are getting *evaluated* at the *time* at which they are evaluated. Assigning a lambda expression to a variable (or reporting a lambda expression from within a custom block) is, of course, possible, even if the lambda expression contains REPORT blocks itself, because at the time of the assignment these REPORT blocks are unevaluated:

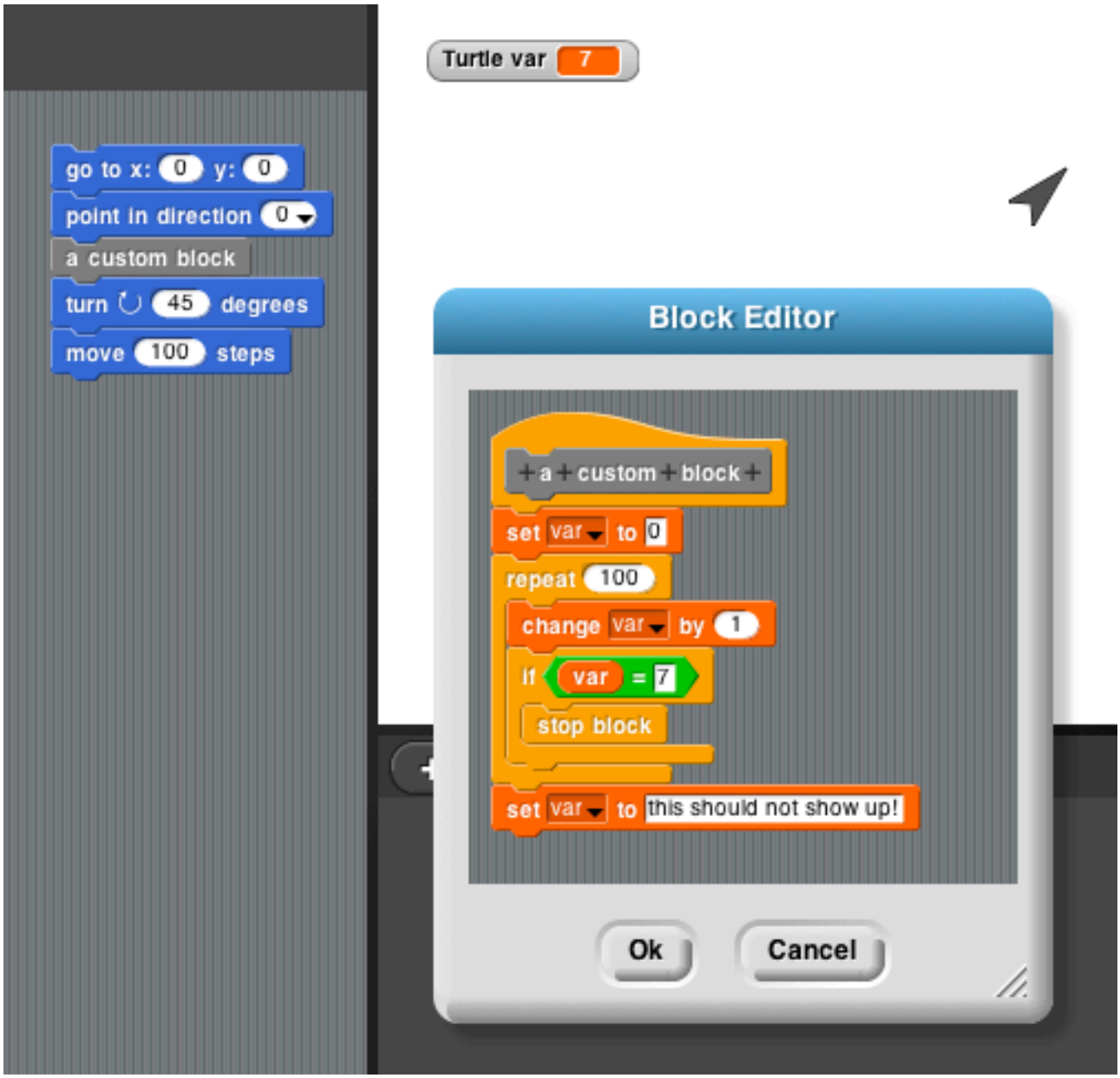


This allows us to write dispatch function style OOP projects in Snap!.

STOP BLOCK Examples

the STOP BLOCK primitive is internally the same as the REPORT primitive, called without an argument.

It is designed to work inside custom block definitions, where it stops the execution of the nearest enclosing custom block definition body, or - in the absence of such - the nearest enclosing lambda expression, or - in the absence of such - the script it is in:



In this exemple, the execution of A CUSTOM BLOCK stops once the Turtle‘s „var“ variable gets set to the number 7. Therefore „var“ is never set to the „this should not show up!“ string. The script enclosing A CUSTOM BLOCK doesn‘t stop and continues to run afterwards, turning and moving the Turtle away from the stage‘s center.

If either the REPORT block or the STOP BLOCK block is used in the regular scripting area, it doesn‘t throw an exception but instead halt the current script, the same as invoking the STOP SCRIPT primitive would: