Article wrapper

Callouts using `co`:

```
(let loopvar ((count 1))
  (if (> count 10)
    #t
    (loopvar (+ count 1))))
```

<calloutlist>
<callout arearefs="dl1">

This variable controls the loop. It is declared without an initial value, immediately after the `let` operand.
</callout>
<callout arearefs="dl2">

Any number of additional local variables can be defined after the loop variable, just as they can in any other `let` expression.
</callout>
<callout arearefs="dl3">

If you ever want the loop to end, you have to put some sort of a test in it.
</callout>
<callout arearefs="dl4">

This is the value that will be returned.
</callout>
<callout arearefs="dl5">

Note that you iterate the loop by using the loop variable as if it was a function name.
</callout>
<callout arearefs="dl6">

The arguments to this function are the values that you want the local variables declared in to have in the next iteration.
</callout>
</calloutlist>