The luashipout package[*]

Marcel Krüger
tex@2krueger.de

April 11, 2020

Overwrite `\shipout` and `\saveboxresource` with a Lua wrapper to add callbacks directly before and after shipout.

TODO: Documentation

1 The implementation

1.1 Lua module

Now we can define our main Lua module:

```
local functions = lua.get_functions_table()
local char_given = token.command_id'char_given'
local token_new = token.new
local make_box_token = token.new(0, token.command_id'make_box')

local orig_shipout = tex.shipout
local orig_saveboxresource = tex.saveboxresource
local setbox = tex.setbox
local put_next = token.put_next
local scan_keyword = token.scan_keyword

local begin_group, end_group do
  local runtoks = tex.runtoks
  local begin_group_t = token.new(0, token.command_id'begin_group')
  local function begin_group_f() return put_next(begin_group_t) end
  function begin_group() return runtoks(begin_group_f) end
  local end_group_t = token.new(0, token.command_id'end_group')
  local function end_group_f() return put_next(end_group_t) end
  function end_group() return runtoks(end_group_f) end
end

local function getclearedbox(b)
  put_next(make_box_token, token_new(b, char_given))
  return token.scan_list()
end
```

[*]This document corresponds to luashipout v0.0.1, dated 2020/04/09.

```
luatexbase.create_callback('pre_shipout_filter', 'list')
luatexbase.create_callback('post_shipout', 'simple')
luatexbase.create_callback('shipout', 'exclusive', function(n, k, ...)
  if k == 'page' then
    begin_group()
      setbox(0, n)
      orig_shipout(0)
    end_group()
  elseif k == 'box' then
    return orig_saveboxresource(n, ...)
  else
    error[[Unsupported shipout type]]
  end
end)

local function generic_shipout(n, ...)
  n = node.direct.is_node(n) or getclearedbox(n)
  local new_n = luatexbase.call_callback('pre_shipout_filter', n, ...)
  if new_n == true then
    new_n = n
  elseif new_n == false then
    node.flush_list(n)
    return
  end
  local index = luatexbase.call_callback('shipout', new_n, ...)
```

In the common case of `kind == 'page'`, `index` will be `nil`. It is only releavant for the `saveboxresource` case.

```
  luatexbase.call_callback('post_shipout', index, ...)
  return index
end

local function shipout(n)
  generic_shipout(n, 'page')
end
tex.shipout = shipout

local function saveboxresource(n, attr, res, imm, t, margin)
  return generic_shipout(n, 'box', attr, res, imm, t, margin)
end
tex.saveboxresource = saveboxresource

local shipout_func =
    luatexbase.new_luafunction'shipout'
local saveboxresource_func =
    luatexbase.new_luafunction'saveboxresource'

functions[shipout_func] = function()
  return shipout(token.scan_list())
```

```
end

functions[saveboxresource_func] = function(imm)
  local t = scan_keyword'type' and token.scan_int()
  local attr = scan_keyword'attr' and token.scan_string()
  local resources = scan_keyword'resources' and token.scan_string()
  local margin = scan_keyword'margin' and token.scan_int()
  local box = token.scan_int()
  saveboxresource(box, attr, res, imm == "immediate", t, margin)
end

token.set_lua('shipout', shipout_func, 'global', 'protected')
token.set_lua('saveboxresource', saveboxresource_func,
              'global', 'protected')
require'luaimmediate'(saveboxresource_func, true)
```

1.2 TEX support package

Most of the time this TEX helper is not needed, but sometimes it can be useful to ensure `\shipout` and `\saveboxresource` are redefined early enough. Especially this has to be loaded *before* `atbegshi`. (Otherwise we effectively disable `atbegshi`)

```
⟨*package⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage
  {luashipout}
  [2020/04/09 v0.0.1 Lua callbacks for shipout]
⟨/package⟩
```

Only LuaLATEX is supported. For other engines we show an error.

```
\ifx\directlua\undefined
⟨*tex-package⟩
  \begingroup
    \ifx\PackageError\undefined
      \def\PackageError#1#2#3{\errhelp{#3}\errmessage{#1: #2}}
    \fi
⟨/tex-package⟩
    \PackageError{luashipout}{LuaLaTeX required}%
    {luashipout requires LuaLaTeX.
     Maybe you forgot to switch the engine in your editor?}
⟨*tex-package⟩
  \endgroup
⟨/tex-package⟩
\fi
```

Load `luaimmediate`. This would normally be loaded by the Lua code directly, but going though TEX allows it to work even if only a `sty` and no `lua` file exists. Of course, the `sty` file would then have to preload the Lua module too.

```
\RequirePackage{luaimmediate}
```

Make sure that `ltluatex` is loaded.

```
\ifx\newluafunction\@undefined
  \input ltluatex
\fi
```

The actual functionality is not that interesting: We just load the Lua module.

```
\directlua{require'luashipout'}
\endinput
```