Literate Programming Example

Version 7.2

March 19, 2020

You are looking at the results of recent experiments with Literate Programming in Racket. The name of the file you're reading is `LPexample.scrbl`. It's a wrapper around the real substance of this program, which is in the included file `LPexample.rkt`.

`LPexample.rkt` is the source file for this document. The Github repository is here.

1 Introduction

Racket's `#lang scribble/lp` and `#lang scribble/lp2` let us write programs in the *Literate Programming Idiom*. Racket's official documentation on `#lang scribble/lp` and `#lang scribble/lp2` is not very clear, however, and that's not surprising: racket's official documentation is sometimes lacking for features outside the core or for topics between the very basic and detailed reference.

You can do two different things with a literate program: *tangle* it or *weave* it. Tangling a literate program means either just running it or spreading its code out on disk in a structure convenient for build tools. Weaving a literate program means rendering it to `HTML`, `LaTeX`, or `PDF`, specifically for human consumption.

Because we want to *run* our programs, we must use the file extension `.rkt`, not `.scrbl`, for literate programs. You can run this here `.rkt`, the one you're reading right now, by typing

```
racket LPexample.rkt
```

at the command line, or by clicking the "Run" button when you have the file open in the DrRacket GUI. It's probably better to run the program in the GUI because the program draws pictures that you can't see in the command line, but it's fine either way.

2 Weaving, or Producing Documentation

Racket's `scribble` command does weaving. You can

```
scribble --pdf LPexample.rkt
```

if you want to do, but the result is ugly. It's much better to wrap this `.rkt` file in a `.scrbl` file written in either `#lang scribble/base` or `#lang scribble/manual`.

The `.scrbl` wrapper can be very basic:

```
#lang scribble/manual
@(require(for-label 2htdp/image))
@require[scribble/lp-include]
@title{Literate Programming Example}
You are looking at the results of recent experiments with Literate
Programming in Racket. The name of the file you're reading is
@code{LPexample.scrbl}. It's a wrapper around the real substance of
this program, which is in the included file @code{LPexample.rkt}.
@lp-include["LPexample.rkt"]
```

To weave `LPexample.rkt`, run the `scribble` command on its wrapper `.scrbl` file.

```
  scribble --pdf LPexample.scrbl
```

makes a `PDF` version, and

```
  scribble --html +m --redirect-main http://docs.racket-lang.org/
LPexample.scrbl
```

makes a `HTML` version.

3 Resolving External References

Notice the first two `require` lines in the `.scrbl` wrapper file.

The first is `(require(for-label 2htdp/image))`, which sets the *documentation phase* references. The *documentation phase* in Racket is analogous to the expansion phase for macros in Racket and other Lisps in that it just focuses on modifying the source code rather than compiling it.

- The first is `(require(for-label 2htdp/image))` which sets the *documentation phase* references. The *documentation phase* in Racket is analogous to the expansion phase for macros in Racket and other Lisps in that it just focuses on modifying the source code rather than compiling it.
- The second step is calling the scribble command with the appropriate flags. The command to weave this document is:

```
scribble --html +m --redirect-main http://docs.racket-lang.org/
LPexample.scrbl
```

4 Tangling

A `scribble/lp` file (`.rkt` extension) contains both the code for *tangling* into a program or library and the text for *weaving* into a document. Like its parent `scribble`, `scribble/lp` allows direct input of text. The code to be tangled is delineated:

```
@chunk[<example_main>
       <example_requires>
       <example_exports>
       <example_body>]
```

Which matches the source for this output from the weaving process:

<***example_main***> **::=**

  <example_importExport>
  <example_body>

Because this is the first

```
@chunk
```

it is treated as the *main chunk*. This is mentioned briefly near the bottom of the `scribble/lp` documentation. If you don't want the first

```
@chunk
```

to serve as the main chunk, then:

```
@chunk[<*>
       <example_requires>
       <example_exports>
       <example_body>]
```

can be placed anywhere in the document to serve as the main chunk. Having tried it, it really doesn't add anything for clarity and is unnecessary.

The reason it is unnecessary is that tangling can entail composing the code in a sequence other than what would normally be used in a `#lang racket` program. For example, required modules need not be near the top. This chunk:

```
@chunk[<example_importExport>
       (require 2htdp/image)
       (provide (all-defined-out))]
```

produces, this output from the weaving process:

<*example_importExport*> **::=**

```
(require 2htdp/image)
(provide (all-defined-out))
```

As shown in the example, source chunks like this:

```
@chunk[<blue_square>
(rectangle 100 100 "solid" "blue")
```

which weaves to this:

<*blue_square*> **::=**

```
(rectangle 100 100 "solid" "blue")
```

can be composed into other functions this way:

```
@chunk[<blue_square>
(beside/align "bottom"
              (ellipse 20 70 "solid" "lightsteelblue")
              <blue_square>)]
```

which weaves out to:

<*example_body*> **::=**

```
(beside/align "bottom"
              (ellipse 20 70 "solid" "lightsteelblue")
              <blue_square>)
```

5 Conclusions

The two items which required teasing out from the documentation are:

- Weaving requires a second file where a file with a `.rkt` file extension is referenced using `lp-include`.
- Tangling treats the first chunk differently unless the `<*>` special name is used.

Happy Literate Programming, Ben.

Update: 13/12/19 - broken links issue fixed and documented thanks to StackOverflow user Asumu Takikawa.