Extended Exercise: Making Web Pages

A Supplement to "How to Design Programs"
(`http://www.htdp.org`)

©2003 Felleisen, Findler, Flatt, Krishnamurthi

1 My First Web Page

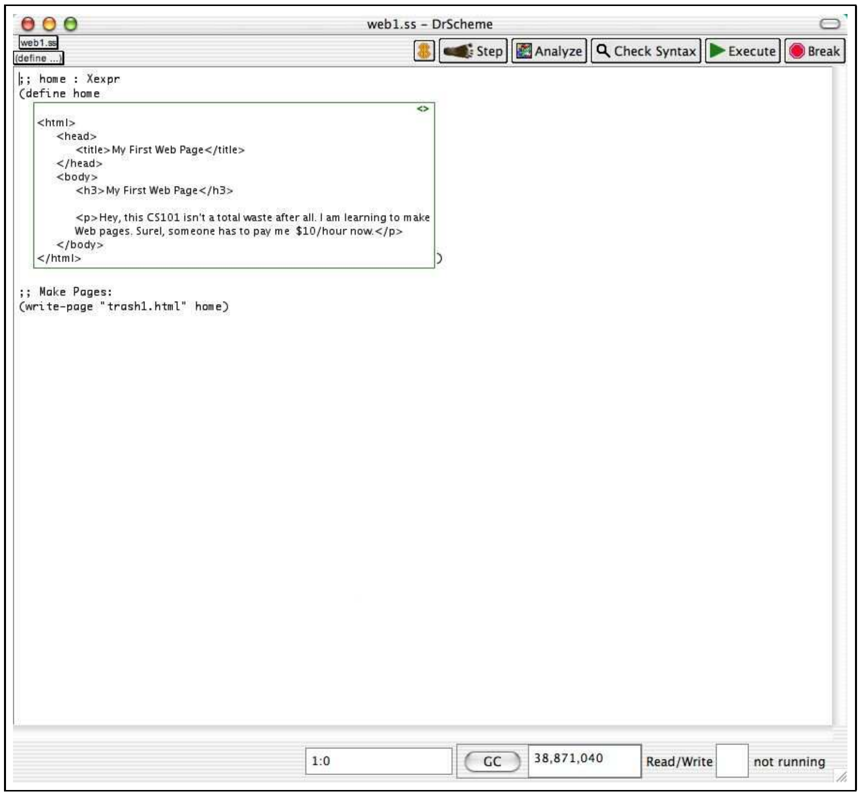
TEACHPACK: webpages.scm

Figure 1 shows how to use DrScheme to produce Web pages in a simplistic manner. The special (green) box is a so-called XML box. Create it via the "special" menu in DrScheme.

Unlike a Web page editing tool, DrScheme requires programmers to type in plain HTML, the programming language of the Web. The figure shows how to produce a trivial page. It's a lot of work. So the question is why should we bother with this when there are much easier ways to produce HTML pages.

This brief essay on Web pages from DrScheme explains why people want to create Web pages with programs with a specific example.

Exercise 1.0.1 The XML boxes are just convenient short-hands for Scheme values, specifically a subset of S-expressions. Load `web1.ss` into DrScheme, add the teachpack, click execute, and find out what these values look like.



```
;; home : Xexpr
(define home
  <>
  <html>
      <head>
          <title>My First Web Page</title>
      </head>
      <body>
          <h3>My First Web Page</h3>

          <p>Hey, this CS101 isn't a total waste after all. I am learning to make
          Web pages. Surel, someone has to pay me  $10/hour now.</p>
      </body>
  </html>
)

;; Make Pages:
(write-page "trash1.html" home)
```

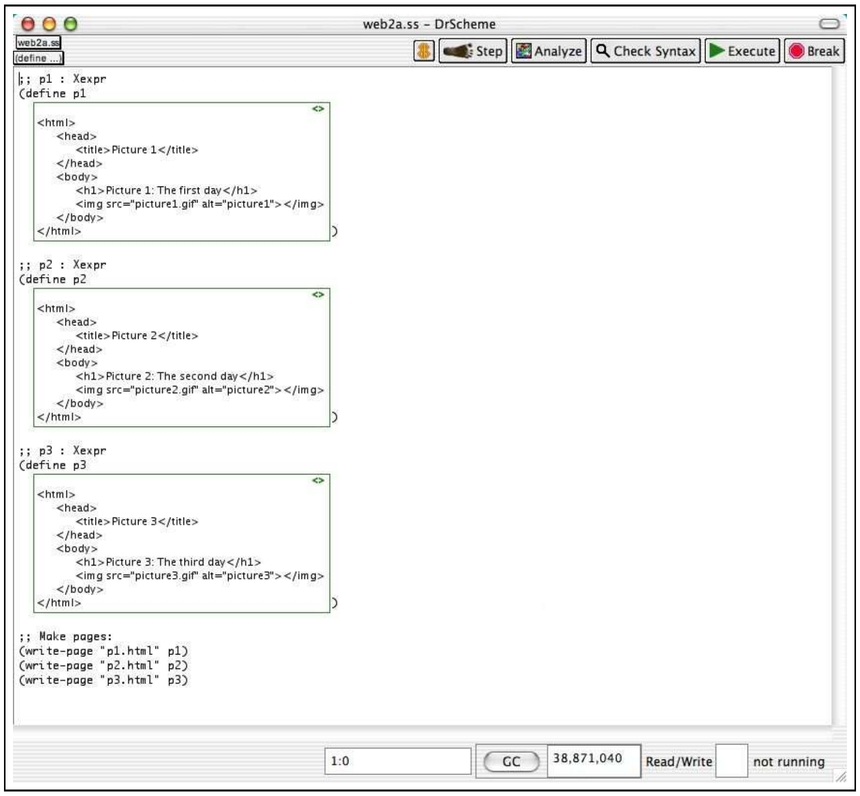
Figure 1: My first Web page

2 My First Program for Making a Web Page

The goal of the section is to understand functions that produce Web pages.

2.1 An HTML solution

Figure 2 shows how to produce a minimalistic photo gallery that consists of three pictures, each displayed on a page by itself.



```
;; p1 : Xexpr
(define p1
  <html>
     <head>
        <title>Picture 1</title>
     </head>
     <body>
        <h1>Picture 1: The first day</h1>
        <img src="picture1.gif" alt="picture1"></img>
     </body>
  </html>
)

;; p2 : Xexpr
(define p2
  <html>
     <head>
        <title>Picture 2</title>
     </head>
     <body>
        <h1>Picture 2: The second day</h1>
        <img src="picture2.gif" alt="picture2"></img>
     </body>
  </html>
)

;; p3 : Xexpr
(define p3
  <html>
     <head>
        <title>Picture 3</title>
     </head>
     <body>
        <h1>Picture 3: The third day</h1>
        <img src="picture3.gif" alt="picture3"></img>
     </body>
  </html>
)

;; Make pages:
(write-page "p1.html" p1)
(write-page "p2.html" p2)
(write-page "p3.html" p3)
```

```
1:0    38,871,040
```

Figure 2:

Exercise 2.1.1 Change the pages in figure 2 so that the sources of the pictures are the jpg files that come with this handout.

Change the pages in figure 2 so that the title of the picture is on the same line, to the left of the picture. Hint: Eliminate the `<h1>` title and use

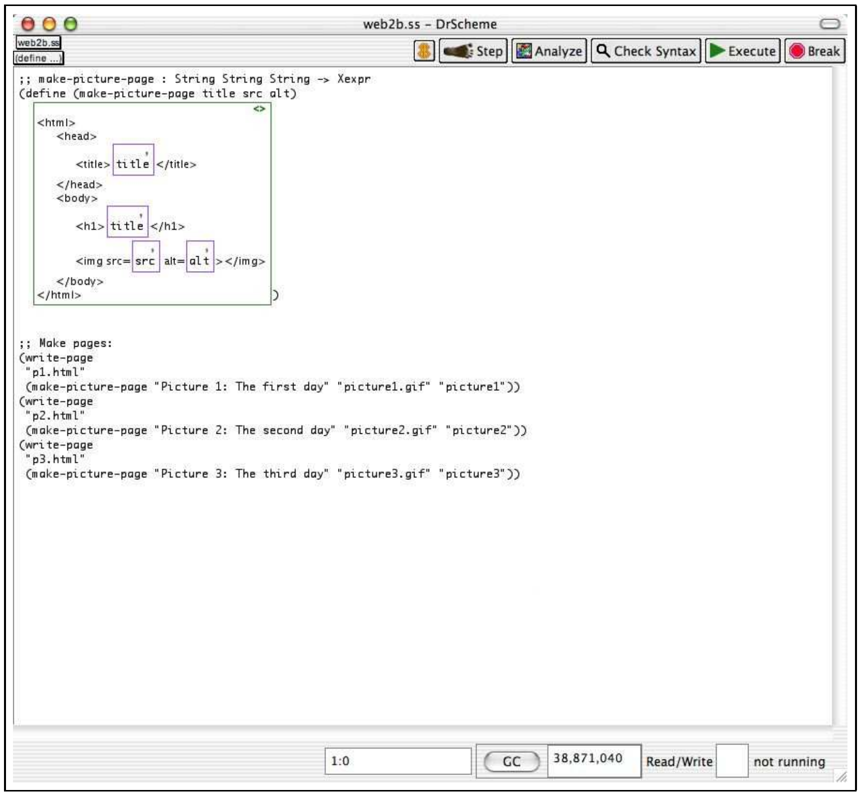
```
<p> ... </p>
```

instead. ∎

2.2 A Scheme solution

Now look at figure 3. It shows how to define programs that produce HTML pages.

The inner boxes are Scheme boxes. Here they just refer to the inputs of the functions and these strings are added in place to the surrounding HTML pages. In general, these Scheme boxes can contain arbitrary Scheme expressions.



```
;; make-picture-page : String String String -> Xexpr
(define (make-picture-page title src alt)
  <html>
    <head>

      <title> title </title>
    </head>
    <body>

      <h1> title </h1>

      <img src= src  alt= alt ></img>
    </body>
  </html>
)

;; Make pages:
(write-page
 "p1.html"
 (make-picture-page "Picture 1: The first day" "picture1.gif" "picture1"))
(write-page
 "p2.html"
 (make-picture-page "Picture 2: The second day" "picture2.gif" "picture2"))
(write-page
 "p3.html"
 (make-picture-page "Picture 3: The third day" "picture3.gif" "picture3"))
```

Figure 3:

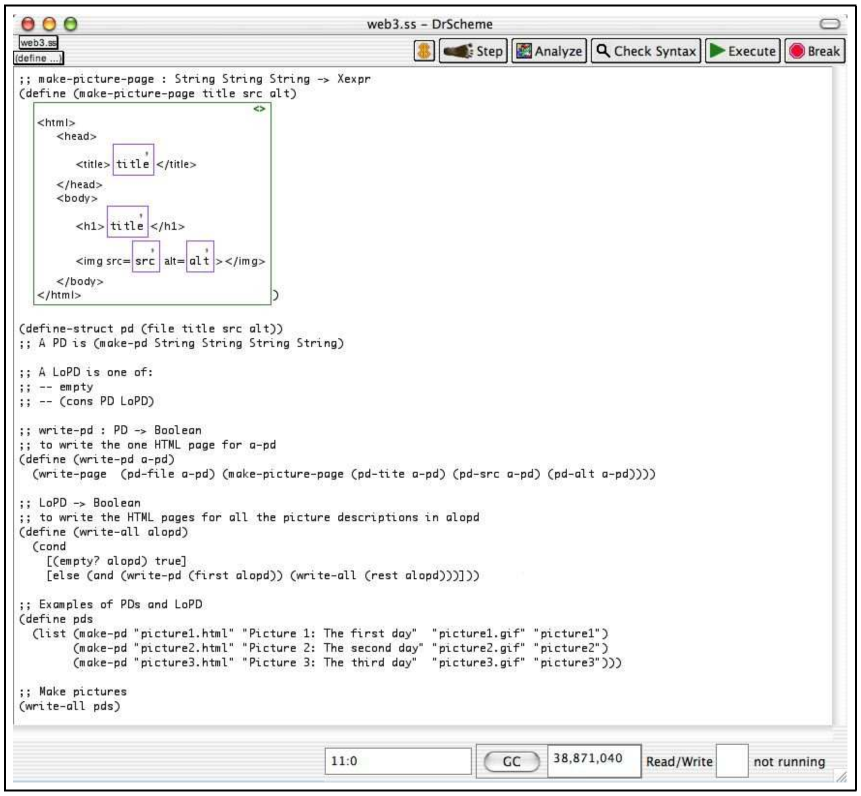
Exercise 2.2.1 Change the pages in figure 3 so that the title of the picture is on the same line, to the left of the picture. Why do we define functions to produce Web pages? Hint: See exercise 2.1.1. ∎

3 Programs for Making a Photo Gallery

A photo gallery doesn't consist of just one or two or three pictures, but dozens. So, we need functions that process lists of photo descriptions. We also need an index file so that we have a central place for all the pictures.

3.1 Photo Descriptions and Lists

Study the program in figure 4.



```
;; make-picture-page : String String String -> Xexpr
(define (make-picture-page title src alt)
  <html>
    <head>
      <title> title </title>
    </head>
    <body>
      <h1> title </h1>
      <img src= src  alt= alt ></img>
    </body>
  </html>
)

(define-struct pd (file title src alt))
;; A PD is (make-pd String String String String)

;; A LoPD is one of:
;; -- empty
;; -- (cons PD LoPD)

;; write-pd : PD -> Boolean
;; to write the one HTML page for a-pd
(define (write-pd a-pd)
  (write-page  (pd-file a-pd) (make-picture-page (pd-tite a-pd) (pd-src a-pd) (pd-alt a-pd))))

;; LoPD -> Boolean
;; to write the HTML pages for all the picture descriptions in alopd
(define (write-all alopd)
  (cond
    [(empty? alopd) true]
    [else (and (write-pd (first alopd)) (write-all (rest alopd)))]))

;; Examples of PDs and LoPD
(define pds
  (list (make-pd "picture1.html" "Picture 1: The first day"  "picture1.gif" "picture1")
        (make-pd "picture2.html" "Picture 2: The second day" "picture2.gif" "picture2")
        (make-pd "picture3.html" "Picture 3: The third day"  "picture3.gif" "picture3")))

;; Make pictures
(write-all pds)
```

Figure 4: Making a photo gallery

Now let's modify it so that we can improve it.

Exercise 3.1.1 If you ignore the file extensions (.html, .gif, .jpg) the file, the src and the alt fields of a PD are the same. The common part is called the *stem*. Use the following PD instead:

(**define-struct** *pd* (*stem title*))

Modify the program accordingly. ∎

Exercise 3.1.2 Design a function that consumes the list of picture descriptions and produces an index file for the gallery. To create an index for three pictures, use `<ul>` ... `</ul>` for making an HTML enumeration and `<li>` ... `<li>` for making items in an enumeration:

```
<ul>
 <li><a href="picture1.html">Picture 1</a></li>
 <li><a href="picture2.html">Picture 2</a></li>
 <li><a href="picture3.html">Picture 3</a></li>
</ul>
```

Design a program to produce the a list of enumeration items from a list of photo descriptions and another one for making wrapping this list with `<ul>` and `</ul>` tags.

Now modify the Web pages for each picture so that they display a link that takes viewers back to the index page. ∎

Exercise 3.1.3 Challenge: Modify the program so that a viewer can also walk through the gallery in a circular way. That is, the viewer starts with the index and then goes to a picture and from there to all the other pictures, one after another. ∎