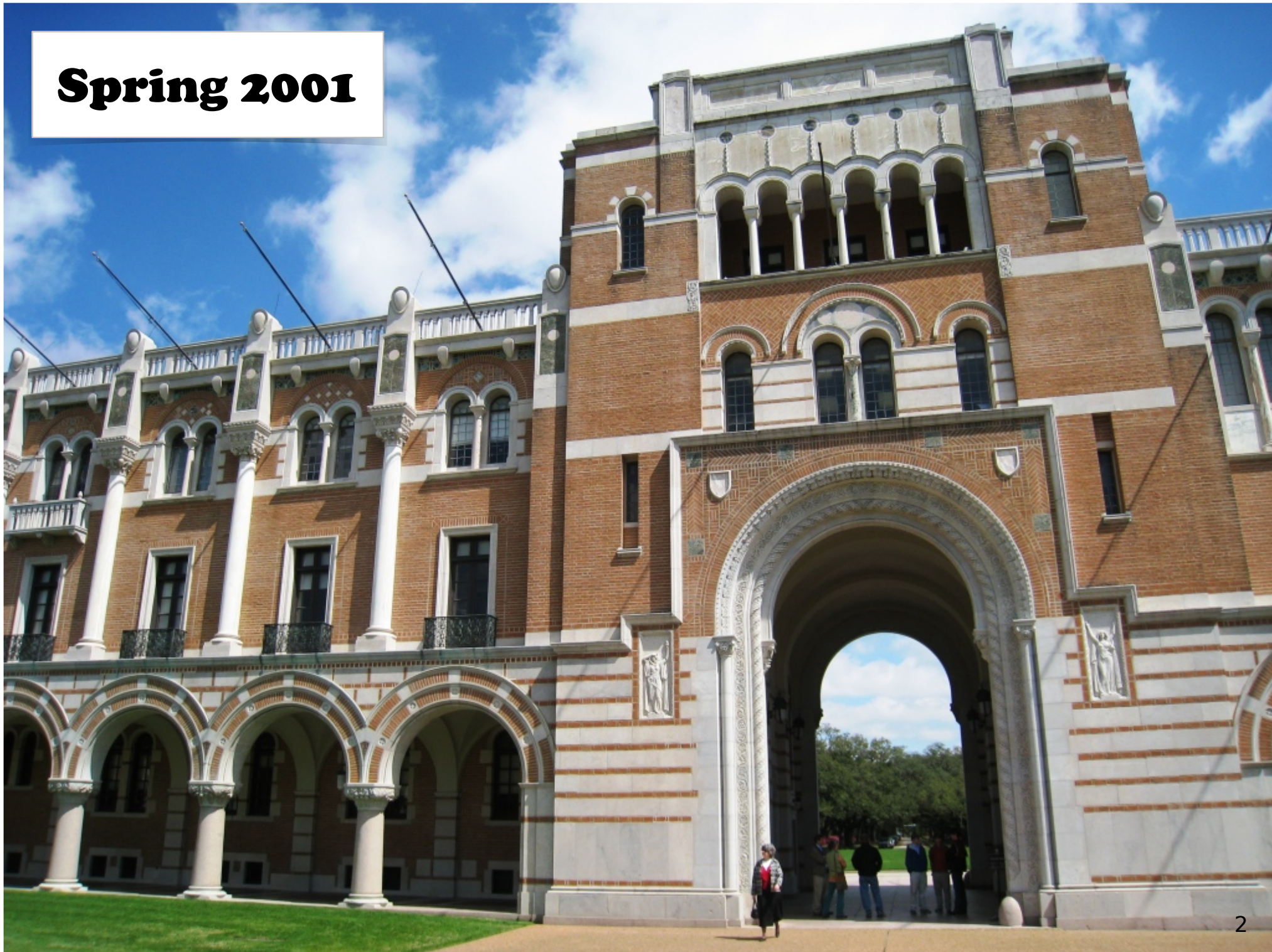


Database Access & Making Low-level Libraries Rackety

**Ryan Culpepper
PLT, University of Utah**

Spring 2001





Matthias Felleisen's Home Page - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.cs.rice.edu/~matthias/

Matthias Felleisen



i-con-o-clast *n.* **1.** One who destroys sacred images. **2.** One who attacks and seeks to overthrow traditional or popular ideas or institutions. *The American Heritage Dictionary* Wordsmith

Q: What is the shortest lie in computing? A: It works.

On Programming: A bad day writing code in Scheme is better than a good day writing code in C. -- David Stigant

More on Programming: Always code as if the guy who ends up maintaining your code will be violent psychopath who knows where you live. -- John F. Woods

On LISP: In twenty years no one will be programming in anything but LISP. -- Piet Hut, IAS Princeton. Who Got Einstein's Office. Ed Regis, pg 166. It's 2001, so there's hope. -- Shriram Krishnamurthi

On the Quality of Shrink Wrap Software: Mark Minasi. [The Software Conspiracy: Why Software Companies Put Out Faulty Products, How They Can Hurt You, and What You Can Do.](#) McGraw-Hill. 1999.



Matthias Felleisen's Home Page - Mozilla Firefox

Comp312: Homework Projects - Mozilla Firefox

File Edit View History Bookmarks Tools Help

http://www.cs.rice.edu/~matthias/312/Homework/

M

F

F

Comp312: Homework Projects

Due Date	Subject
Jan 23	Creating Web Content
Jan 30	Processing XML
Feb 06	Creating Web Content Dynamically
Feb 13	Processing Web Content
Feb 20	Link Checking Web Sites
Feb 27	A Gas Station Simulator
Mar 13	Extending the Gas Station Simulator
Mar 20	A New Look for the Gas Station Simulator
Apr 03	Correctness Proofs
Apr 10	Using Tools
Apr 17	Building a Web Server
Apr 24	Extending a Web Server

[312 Home](#)

Powered by [PLT Scheme](#)

The screenshot shows a Mozilla Firefox browser window with the following details:

- Window title: Matthias Felleisen's Home Page - Mozilla Firefox
- Tab title: Comp312: Homework Projects - Mozilla Firefox
- Page title: Creating Web Contents Dynamically - Mozilla Firefox
- Address bar: <http://www.cs.rice.edu/~matthias/312/Homework/editorial.html>

The page content includes:

Creating Web Contents Dynamically

Your task is to develop a CGI script that helps people order a political commentary of their liking. The dialog consists of three steps:

- a welcome page that asks for the login name, the password, and the current preference;
- a page that displays the last four digits of the credit cards used in the past and that gives the consumer the option of submitting a new credit card number; and
- a confirmation page that summarizes the dialog.

Consider the following example:

welcome credit card choice goo bye

The color of the page changes as the consumer progresses through the dialogue.

If the consumer choose to use a new credit card number, the program must know about the new number when the consumer logs in again. You may store the login names, passwords, and credit card numbers in a file. Do not worry about security concerns.

[312 Hom](#) If at any point in the dialog something goes wrong, the program should display a red page with white text:

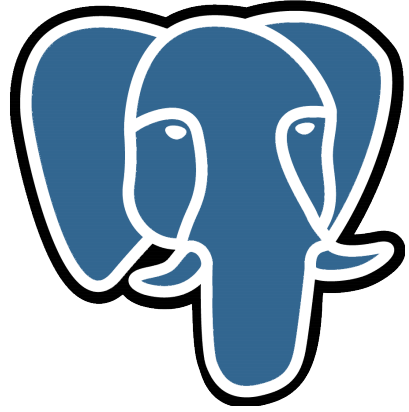
warning

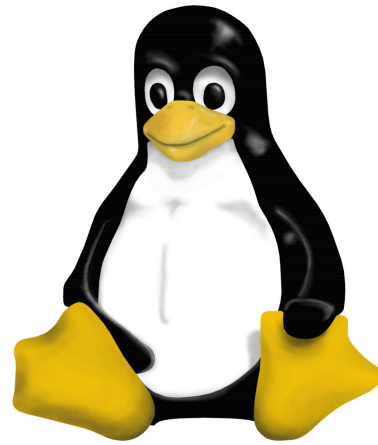
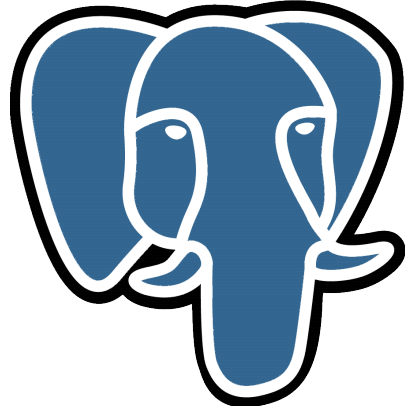
The warning here explains what the consumer might have done wrong and provides a button for starting over.

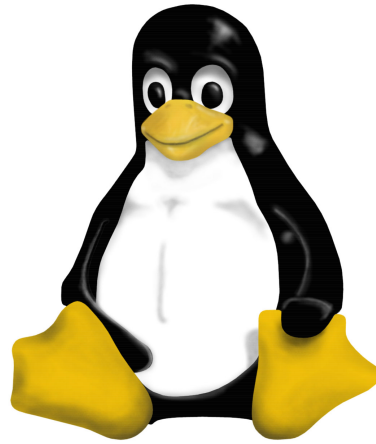
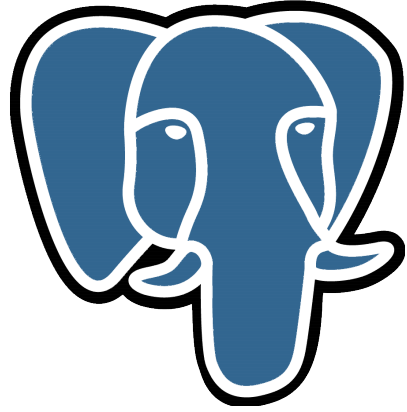
Notes:

- You program must be started with a URL that runs to a cgi script.
- You are free to choose the exact words, the exact layout of the page, the colors (except for red), etc.
- Don't overstep your freedom. Your program must conform to the functional specifications stated above.

[Matthias Felleisen](#)









```
<?php
```

```
// String, String, String --> String
function handle_payment_main($username, $password, $wing) {
    $db = pg_connect("", "", "", "", "comp312");

    $login_sql = <<<END_SQL
        select password from users
        where username = '$username';
    END_SQL;

    $login_query = pg_exec($db, $login_sql);
    $login_numrows = pg_numrows($login_query);

    if ($login_numrows == -1) {
        // Error.
        echo "Whoops, error on query <b>$sql</b>";
        exit();
    }
    if ($login_numrows == 0) {
        // The user was not found. We will add them and continue.
        add_user($db, $username, $password);
    } else {
        // Get the (sole) row
        $login_row = pg_fetch_array($login_query, 0);
        $real_password = $login_row['password'];

        if ($password == $real_password) {
            // Okay. Continue with script.

```

Q:

Why did you do this in ?

Q:

Why did you do this in ?

A:

Because a database was the right tool for the job.

Q:

Were you aware that **PLT**  has database support?



```
(define henv
  (alloc-handle 'sql-handle-env))
(set-env-attr henv
  'sql-attr-odbc-version
  'sql-ov-odbc3)

(define hdbc
  (alloc-handle 'sql-handle-dbc henv))
(connect hdbc datasource user passwd)

(define hstmt
  (alloc-handle 'sql-handle-stmt))
(exec-direct hstmt query-sql)
(fetch hstmt)
(get-data hstmt 1 buffer indicator)
...
(free-handle hstmt)
(free-handle hdbc)
```

```
(define henv
  (all-procedure-handle-procedure-args))
```

PLT 

```
(set
```



```
(def
```

```
(a
```

```
(con
```

```
(def
```

```
(a
```

```
(exe
```

```
(fet
```

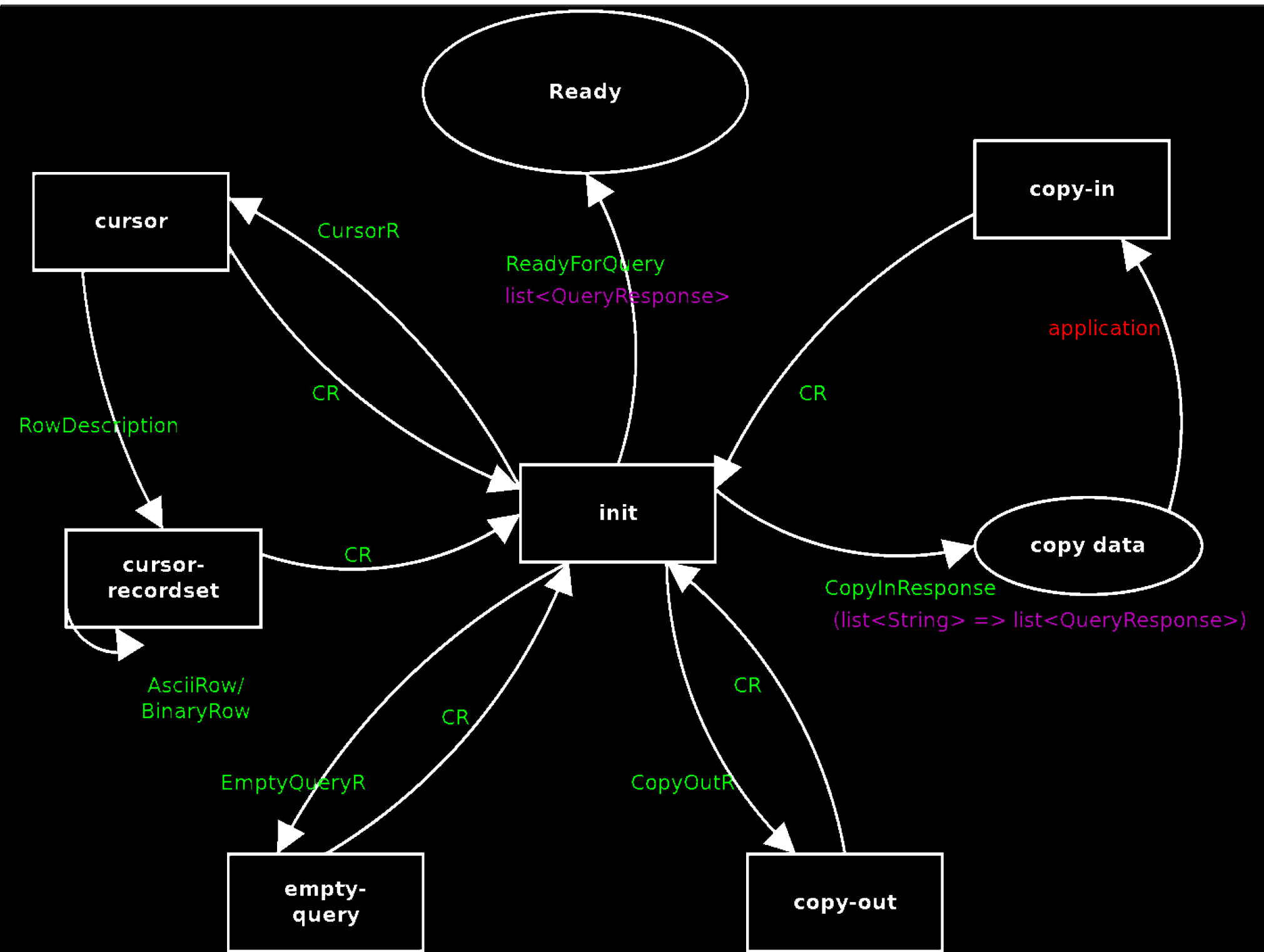
```
(get
```

```
...
```

```
(fre
```

```
(fre
```

```
$db = pg_connect("", "", "", "", "comp312");
$login_query = pg_exec($db, $login_sql);
$login_numrows = pg_numrows($login_query);
$login_row = pg_fetch_array($login_query, 0);
```

spgsql

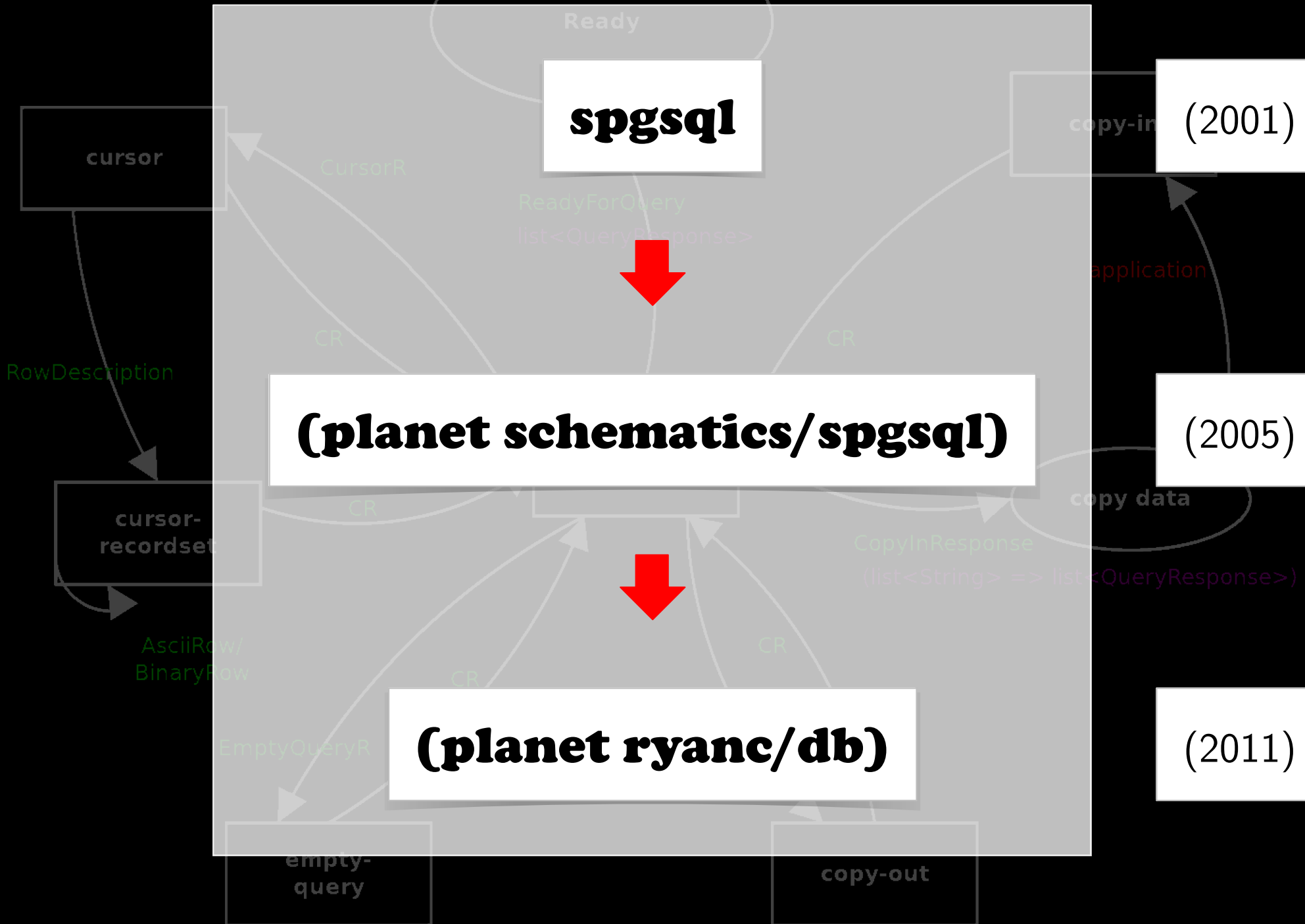
(2001)

(planet schematics/spgsql)

(2005)

(planet ryanc/db)

(2011)

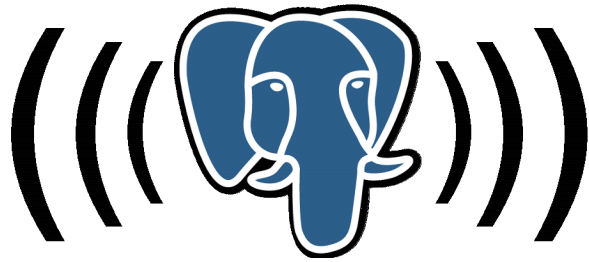


in defense of SrPersist

based mature interface

existing drivers

PLT  version 103



the db library
**A database interface for
functional programmers**

PostgreSQL



ODBC

PostgreSQL



wire protocol

SQLite 

ODBC

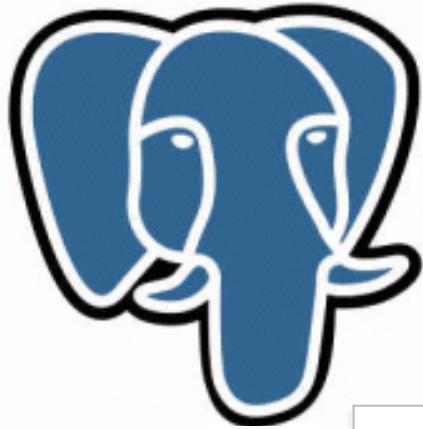
PostgreSQL



ODBC

ffi

PostgreSQL



base connections



ODBC


```
(postgresql-connect  
  #:database "mydb"  
  #:user "ryan"  
  #:password "secret")
```

```
(postgresql-connect
  #:database "mydb"
  #:user "ryan"
  #:password "secret"
  #:server "db.mysite.com"
  #:port 5432
  #:ssl 'yes
  #:notice-handler (current-output-port))
```

```
(postgresql-connect  
  #:database "mydb"  
  #:user "ryan"  
  #:password "secret"  
  #:socket  
  "/var/run/postgresql/.s.PGSQL.5432")
```

```
(postgresql-connect  
  #:database "mydb"  
  #:user "ryan"  
  #:password "secret"  
  #:socket 'guess)
```

```
(mysql-connect #:database "mydb"  
              #:user "ryan"  
              #:password "secret")
```

```
(sqlite3-connect #:database "~/my.db")
```

```
(odbc-connect #:dsn "mydsn")
```

```
(put-dsn 'pg
        (postgresql-data-source
          #:database "mydb"
          #:user "ryan"
          #:password "secret"))
```

```
(dsn-connect 'pg)
```

```
(dsn-connect 'pg #:ssl 'yes)
```

```
create table the_numbers (n integer, d varchar(20))
```

```
> (query c "select n, d from the_numbers \  
          where n < 3")  
  
(recordset  
  '(((name . "n") ....) ((name . "d") ....))  
  (list (vector 0 "zero")  
        (vector 1 "one")  
        (vector 2 "two")))
```

```
create table the_numbers (n integer, d varchar(20))
```

```
> (query-exec c "delete from the_numbers \  
                where n = 0")
```

```
> (query-rows c "select n, d from the_numbers \  
                where n < 3")
```

```
(list (vector 1 "one") (vector 2 "two"))
```



```
create table the_numbers (n integer, d varchar(20))
```

```
> (query-value c "select d from the_numbers \  
                    where n = 1")
```

```
"one"
```

```
> (query-list c "select n from the_numbers \  
                    where n < 3")
```

```
'(1 2)
```

```
> (query-row c "select n, d from the_numbers \  
                    where n % 5 = 0")
```

```
(vector 5 "five")
```

```
create table the_numbers (n integer, d varchar(20))
```

```
> (in-query c "select n, d from the_numbers \  
      where n <= 3")  
  
#<sequence>  
  
> (for ([(n d)  
      (in-query c "select n, d from the_numbers \  
      where n <= 3")])  
      (printf "~a: ~a\n" n d))  
  
1: one  
2: two  
3: three
```

```
create table the_numbers (n integer, d varchar(20))
```

```
> (query-value pg-c  
    "select d from the_numbers \  
    where n = $1"  
    7)  
"seven"
```

```
create table the_numbers (n integer, d varchar(20))
```

```
> (define get-d-between  
    (prepare my-c  
        "select d from the_numbers \  
        where n > ? and n < ?"))  
  
> (query-list my-c get-d-between 1 6)  
(list "two" "three" "four" "five")  
  
> (query-value my-c get-d-between 0 2)  
"one"
```

Non-goal:

Abstracting over different SQL dialects (even placeholder syntax)

SQL

varchar, etc

integer, etc

numeric, decimal

real

blob, bytea

Racket

string

exact integer

exact rational

inexact real

bytes

```
(struct sql-date (year month day))
```

```
(struct sql-time (hour minute second nanosecond  
tz))
```

```
(struct sql-timestamp (year month day  
hour minute second nanosecond  
tz))
```

```
(struct sql-interval (years months  
days hours minutes  
seconds nanoseconds))
```

Other SQL types

- bit vectors
- geometry

A problem:

Web servlet database connection lifetimes

- short
- unpredictable

connection pool

```
(define pool
  (connection-pool (lambda () ....)))

(let ([c (connection-pool-lease pool)])
  .... do work with c ....
  (disconnect c))
```

virtual connection

```
(define c  
  (virtual-connection (lambda () ....)))  
  
.... do work with c ....
```

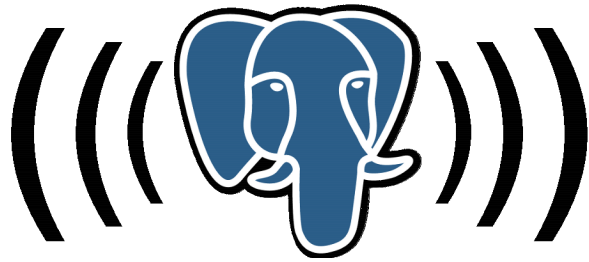
```
(define pool (connection-pool (lambda () ....)))

(define c
  (virtual-connection
    (lambda ()
      (connection-pool-lease pool
                              (current-thread))))))

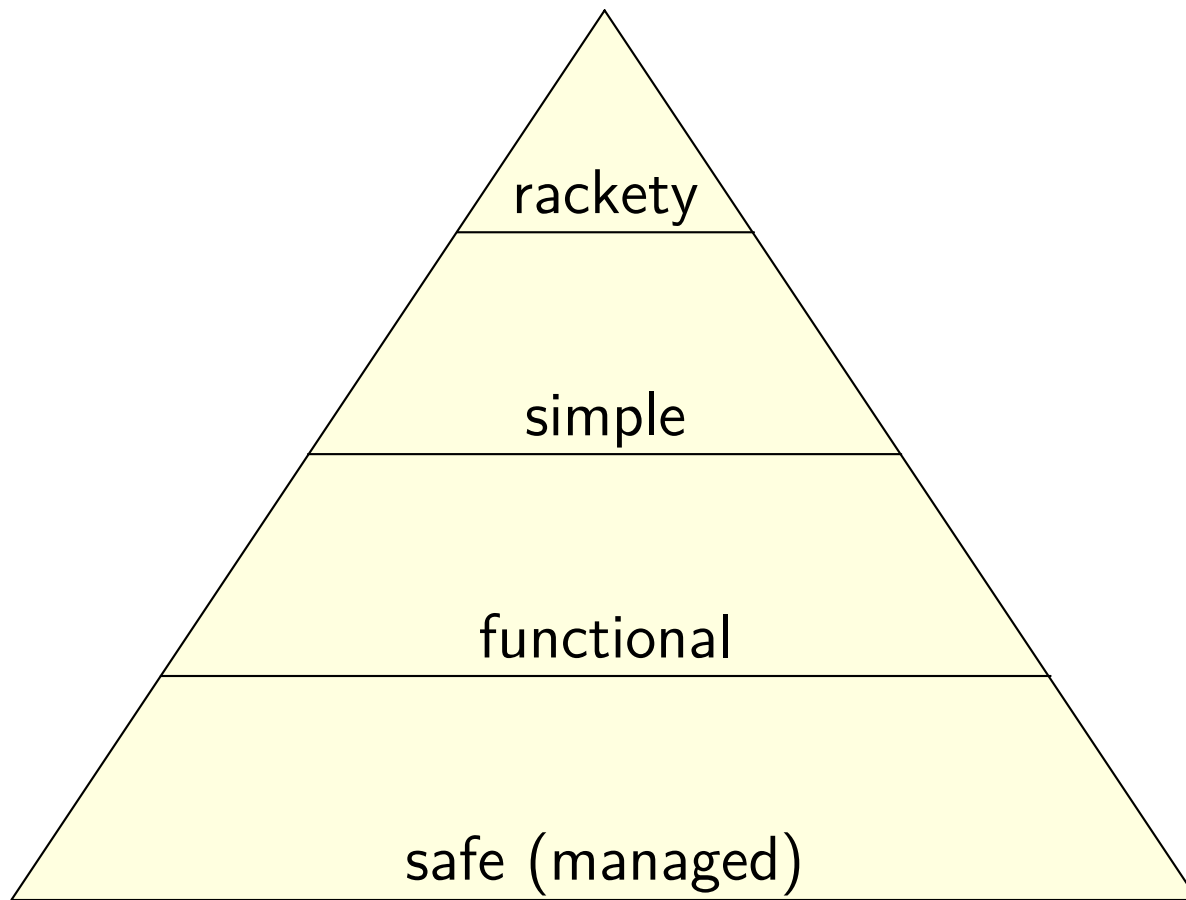
.... do work with c ....
```

```
(define c
  (virtual-connection
    (connection-pool (lambda () ....))))

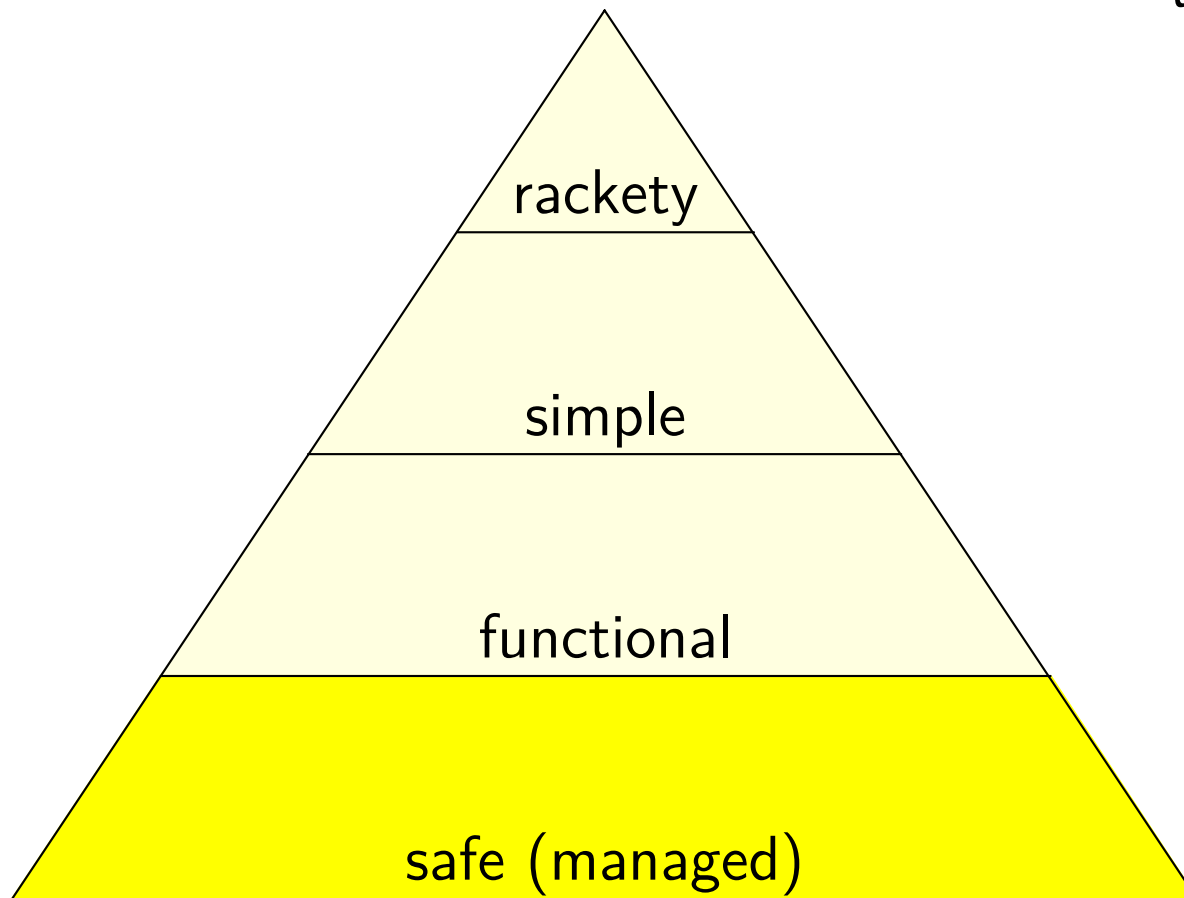
.... do work with c ....
```



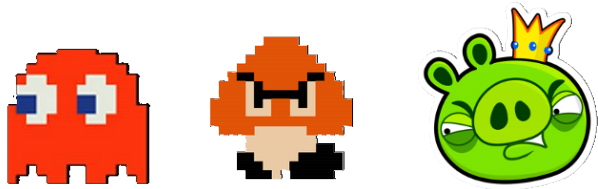
**making libraries
rackety**



no crashes
manage resources (GC)
thread-safe, etc



no crashes
manage resources (GC)
thread-safe, etc

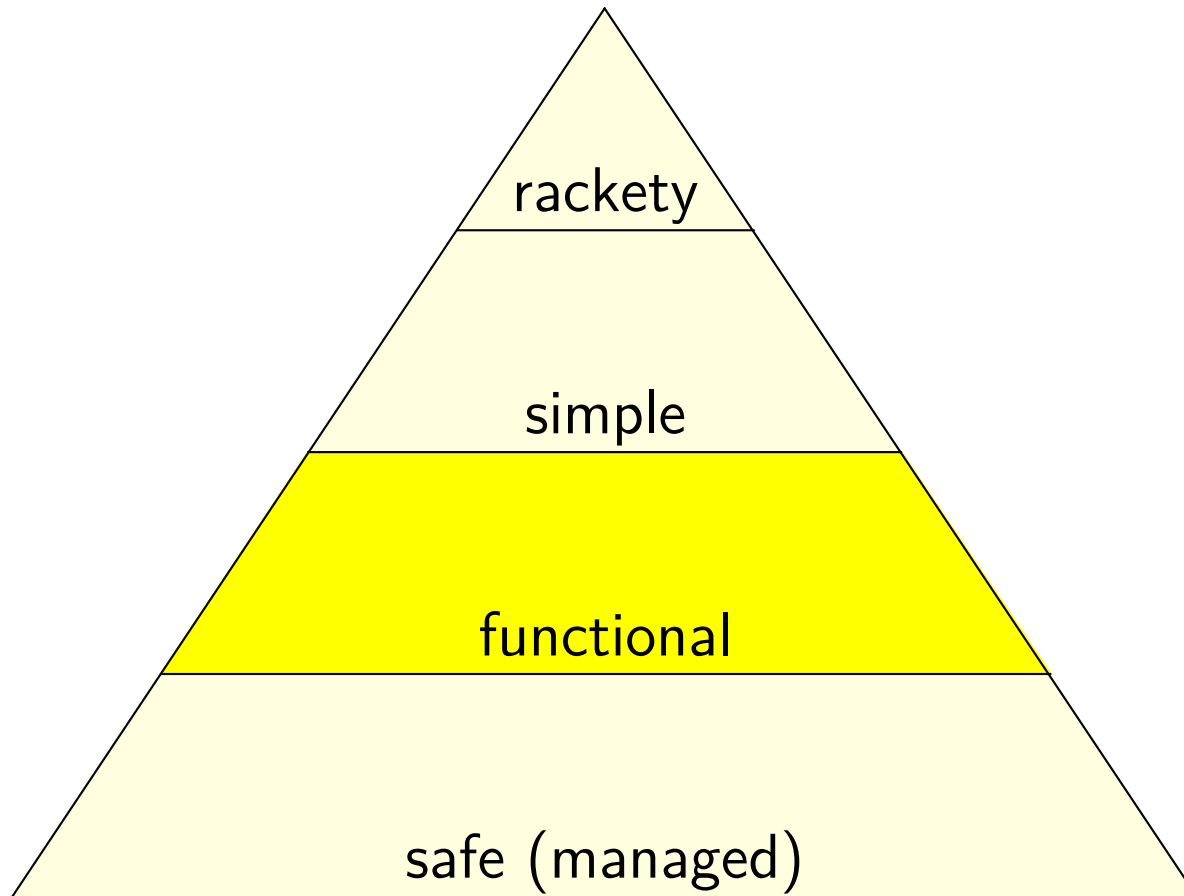


- continuations, exceptions
- break-thread, kill-thread
- custodian-shutdown-all

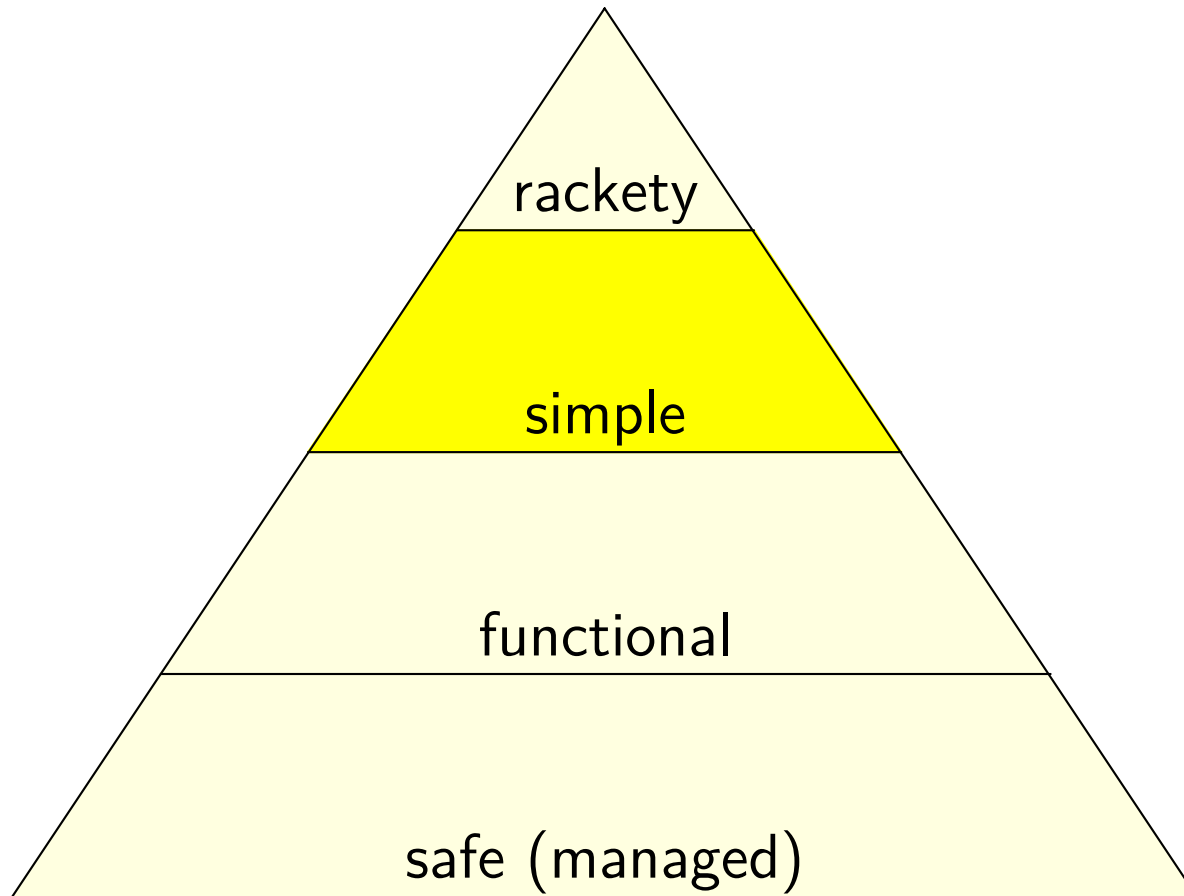
functional

safe (managed)

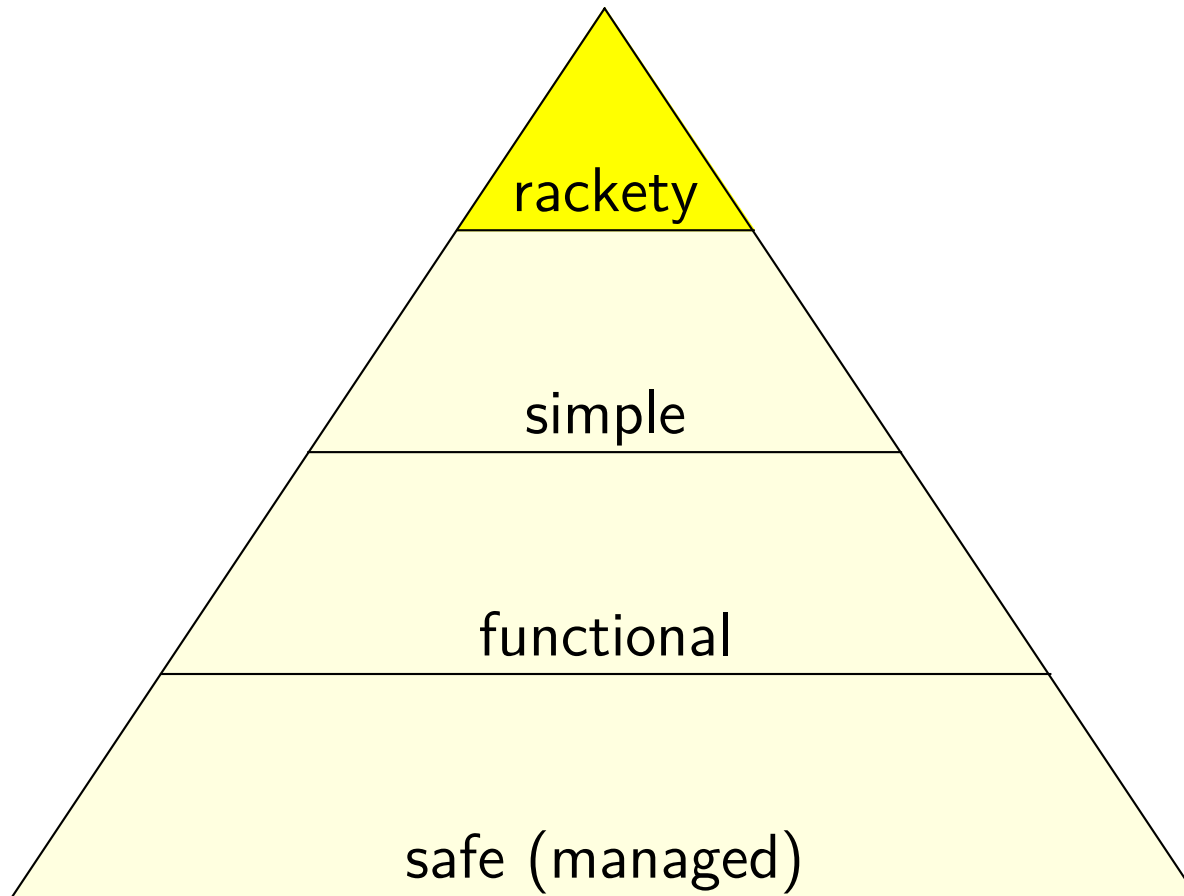
eliminate superfluous side-effects



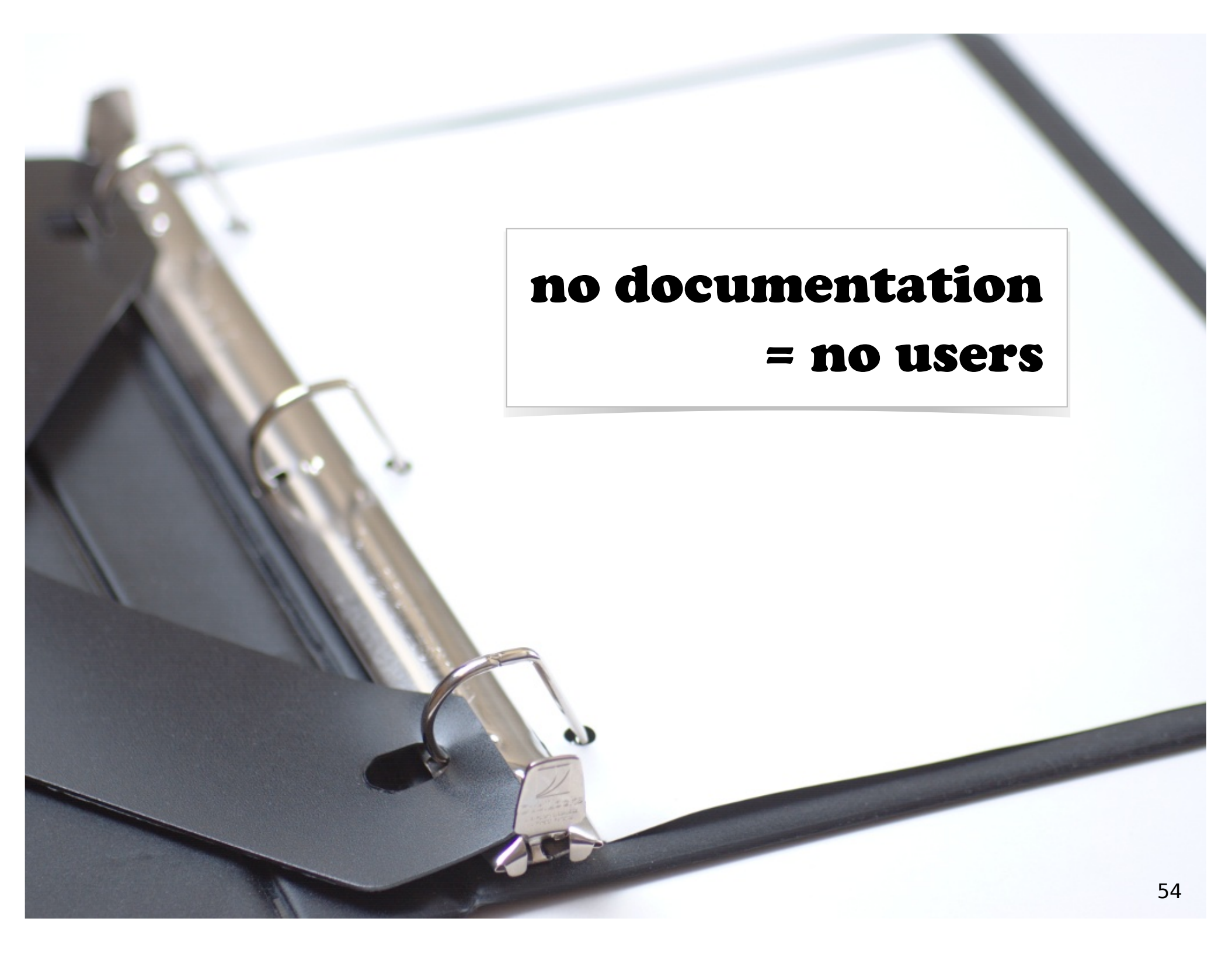
eliminate superfluous concepts



cooperate with Racket features







**no documentation
= no users**

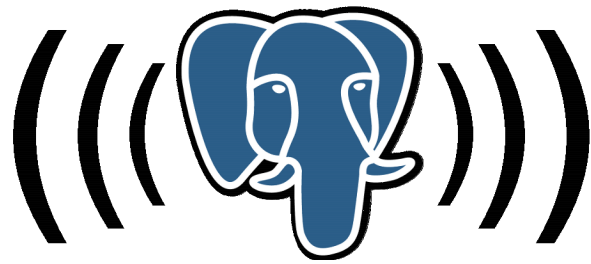


**writing documentation shames us
into removing **little idiocies**
from our designs**

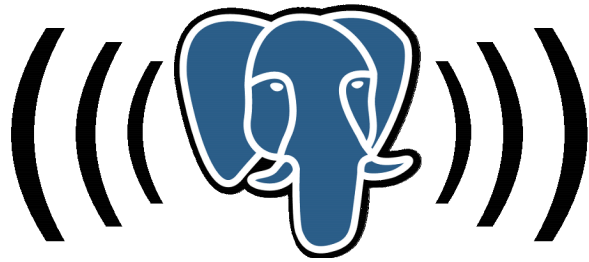


**writing documentation shames us
into removing **little idiocies**
from our designs**

**if you can't explain it,
redesign it**



Questions?



Questions

I have them for you.

ursors

date/time

geometry