Reactive Functional Programming
(or, Is That a Parenthesis in Your Pocket?)

Shriram Krishnamurthi
Pascal
“Pascal is part of the same machinery as hall passes, dress codes, advisors’ signatures, single-sex dorms, and so on.”

—Brian Harvey
The Flow of Control

The **world** is in charge, not the program.
flights from PVD to CDG
must remember
list of flights
list of flights
the third outward flight
must remember
restricted list of return flights
restricted list of return flights
the second return flight
ok
user moved right 10 yards

ok

user tilted phone left

ok

user moved left 10 yards

ok
Elapsed Time:

29
var timerID = null;
var elapsedTime = 0;

function doEverySecond() {
    elapsedTime += 1;
    document.getElementById('curTime').innerHTML = elapsedTime;
}

function startTimer() {
    timerId = setInterval(doEverySecond, 1000);
}

function resetElapsed() {
    elapsedTime = 0;
}

<body onload="startTimer()">
<input id="reset" type="button" value="Reset" onclick="resetElapsed()" />
<div id='curTime'>  </div></body>
void is a Four-Letter Word
(define gauge
  (new gauge% [label "Elapsed"]
              [range 150]))

(new timer%
  [interval 100]
  [callback (lambda (event)
              (send gauge set-value
                     (add1 (send gauge get-value))))])

(new button%
  [label "Reset"]
  [callback (lambda (event)
             (send gauge set-value 0))])
(define gauge
  (new gauge% [label "Elapsed"]
    [range 150]))

(new timer%
  [interval 100]
  [callback (lambda (event)
                (send gauge set-value
                  (add1 (send gauge get-value))))])

(new button%
  [label "Reset"]
  [callback (lambda (event)
             (send gauge set-value 0))])
interface ChangeListener extends EventListener {
    void stateChanged(ChangeEvent e) { ... } }

interface ActionListener extends EventListener {
    void actionPerformed(ActionEvent e) { ... } }

interface MouseListener extends EventListener {
    void mouseClicked(MouseEvent e) { ... }
    void mouseEntered(MouseEvent e) { ... }
    void mouseExited(MouseEvent e) { ... }
    void mousePressed(MouseEvent e) { ... }
    void mouseReleased(MouseEvent e) { ... } }
interface ChangeListener extends EventListener {
    void stateChanged(ChangeEvent e) { ... } }

interface ActionListener extends EventListener {
    void actionPerformed(ActionEvent e) { ... } }

interface MouseListener extends EventListener {
    void mouseClicked(MouseEvent e) { ... }
    void mouseEntered(MouseEvent e) { ... }
    void mouseExited(MouseEvent e) { ... }
    void mousePressed(MouseEvent e) { ... }
    void mouseReleased(MouseEvent e) { ... } }
mainLoop : unit -> unit

closeTk : unit -> unit

destroy : 'a Widget.widget -> unit

update : unit -> unit

pack : ... -> 'd Widget.widget list -> unit

grid : ... -> 'b Widget.widget list -> unit

raise_window : ?above:'a Widget.widget
                        -> 'b Widget.widget -> unit

bind : events:event list
            -> 'a Widget.widget -> unit
mainLoop : unit -> unit
closeTk : unit -> unit

destroy : 'a Widget.widget -> unit
update : unit -> unit

pack : ... -> 'd Widget.widget list -> unit
grid : ... -> 'b Widget.widget list -> unit

raise_window : ?above:'a Widget.widget
  -> 'b Widget.widget -> unit
bind : events:event list
  -> 'a Widget.widget -> unit
propagateEvent :: IO ()
select :: Selecting w => Event w (IO ()
mouse :: Reactive w => Event w (EventMouse -> IO ()
keyboard :: Reactive w =>
  Event w (EventKey -> IO ()
resize :: Reactive w => Event w (IO ()
focus :: Reactive w => Event w (Bool -> IO ()
activate :: Reactive w => Event w (Bool -> IO ()
enter :: Reactive w => Event w (Point -> IO ()
leave :: Reactive w => Event w (Point -> IO ()
motion :: Reactive w => Event w (Point -> IO ()
  drag :: Reactive w => Event w (Point -> IO ()
  click :: Reactive w => Event w (Point -> IO ()
unclick :: Reactive w => Event w (Point -> IO ()
doubleClick :: Reactive w =>
  Event w (Point -> IO ()
  }
propagateEvent :: IO ()
select :: Selecting w => Event w (IO ()
mouse :: Reactive w => Event w (EventMouse -> IO ()
keyboard :: Reactive w =>
    Event w (EventKey -> IO ()
resize :: Reactive w => Event w (IO ()
focus :: Reactive w => Event w (Bool -> IO ())
activate :: Reactive w => Event w (Bool -> IO ()
enter :: Reactive w => Event w (Point -> IO ()
leave :: Reactive w => Event w (Point -> IO ()
motion :: Reactive w => Event w (Point -> IO ()
drag :: Reactive w => Event w (Point -> IO ()
click :: Reactive w => Event w (Point -> IO ()
unclick :: Reactive w => Event w (Point -> IO ()
doubleClick :: Reactive w =>
    Event w (Point -> IO ()

So What?

We care deeply about functions because of Testing Composition Education
Solutions

Lazy evaluation
(Haskell FRP)
Continuations
(PLT Scheme Web Server)
Dataflow + state + GUIs in call-by-value
(FrTime, Flapjax, …)
Why **void**?

The hostile operating system

Why hostile?

Because it’s neutral

But neutrality ≠ hostility

Neutrals can cooperate (impartially)
The information you need to restore the computation from a crash
What’s changing?

*The location of the UFO.*

How do we represent it?

*As a coordinate pair.*
(define width 400)
(define height 300)
(define initial-world 0)
(define PLANE img)

world → scene
(define (render w)
  (place-image PLANE
      (* w 10)
      (image-height PLANE)
      (empty-scene width height)))

world → world
(define (incr-time w)
  (add1 w))

(big-bang initial-world
  (on-tick incr-time)
  (to-draw render))
(define width 400)
(define height 300)
(define initial-world 0)
(define PLANE

\[world \rightarrow \text{scene}\]
(define (render w)
  (place-image PLANE
    (* w 10)
    (image-height PLANE)
    (empty-scene width height)))

(animate initial-world render)

Functions as arguments (sssh!)
on-click :: $w \times btn \rightarrow w$

on-key :: $w \times key \rightarrow w$

on-tilt :: $w \times incl \rightarrow w$

on-move :: $w \times dist \rightarrow w$

to-draw :: $w \rightarrow scene$

on-tick :: $w \rightarrow w$

stop-when :: $w \rightarrow bool$
(define WIDTH 300)
(define HEIGHT 300)

(define-struct vel (x y))

(define target (make-posn (random WIDTH) ...))

(define-struct world (posn r vel))

(define initial-world
  (make-world (make-posn ...) 30 (make-vel 0 0)))

world \rightarrow \text{bool}

(define (game-ends? w)
  (or (\leq (world-r w) 1)
      (collide? w)))
world → world
(define (tick w)
  (make-world (posn+vel (world-posn w) (world-vel w))
               (- (world-r w) 1/3)
               (world-vel w)))

world × number × number × number → world
(define (tilt w azimuth pitch roll)
  (update-world-vel w
                   (update-world-vel w
                        (make-vel roll (- pitch)))))

(big-bang initial-world
  (to-draw render)
  (on-tick tick)
  (on-tilt tilt)
  (stop-when game-ends?))
TOUR GUIDE

No doubt you remember the campus tour—the walking backwards, the explanation of meal plans, the intramural sports speech... It's a fine tradition, but the admissions department is running out of willing volunteers. That's where you come in—you've been asked to write a virtual tour guide that will run on a visitor's cell phone. Users can select the type of destinations they wish to see and receive real-time directions from their current location to the nearest destination.

Specifically, the Admissions Office wants an application with the following features:

- Your program should run on a mobile phone and give dynamically updated tour directions based on its current location relative to a map supplied by the Admissions Office (see below for the exact data definition).

- The user should be able to pick from predefined 'tours', such as a 'Campus Art Tour' or a 'Freshman Dorm Tour'. These tours will also be supplied by the Admissions Office.
2006 PISA scores: USA not in top 20 in math, science, or reading

“Economic Time Bomb”
—June Kronholz, *WSJ*
Salient Curricular Facts

STEM “sorting” occurs in middle-school

Functions are a major barrier

Algebra only over numbers is boring
A train leaves Chicago at 6pm, traveling east at 50mph...
(place-image (star 30 "solid" "white")
  330 180
  (place-image (circle 120 "solid" "red")
    360 165
    (place-image (circle 90 "solid" "white")
      300 210
      (rectangle 600 420 "solid" "red")))
Early Exercise with Images: Making Fabric and Clothing

© Kathi Fisler, 2004-2009

This exercise has students create images of fabrics, create clothing items from fabrics, and put logos onto clothing items. It is designed to give students practice composing functions.

Prerequisites: Numbers, Functions, Images, Defining Constants (Section 3.2)
5 + 3

17 = □ - 11

\[ x^2 = 25 - 16 \]

\[ 17 + 33 = 3x - 20 \]

\[ 1729 = x^3 - 1000 \]

\[ f(x) = x^2 \]

- with attributes
- with multiple representations
Functions

\[ f(x) = x^2 \]

<table>
<thead>
<tr>
<th>x</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>f(x)</td>
<td>1</td>
<td>4</td>
<td>9</td>
<td>16</td>
<td>25</td>
<td>36</td>
</tr>
</tbody>
</table>
(define (f x) (* x x))

(f (+ 3 4))  ->  (f 7)

(define (f x) (* x x))

(f 7)  ->  (* 7 7)

(define (f x) (* x x))

(* 7 7)  ->  49
(define (f x) (* x x))
(define (g x) (* x x x x))

(check-expect (f 1) 1)
(check-expect (f 2) 4)
(check-expect (f 3) 9)
(check-expect (f 4) 16)

Welcome to DrRacket, version 5.1 [3m].
Language: Beginning Student [custom];
memory limit: 128 MB.
Teachpack: image.ss.
All 4 tests passed!
Vital Statistics

Middle-school, after-school (predominantly)
All teaching by volunteers (professionals, students)
Day 1: design your own game; 9 weeks: implement it

24% female; 70% reporting race are minority;
70% on free or reduced-price lunch

Emphasis on: testing, pair-programming, code-review
Over 700 students (average age: 11y9m) so far
About 200 new students this past year, growing /year
Technology

Scheme (Racket) on the desktop
Scheme → J2ME compiler for Android
Scheme → JavaScript
  Works on both phones and browsers
  Need to deal with asynchrony
  Need to deal with re-entrance
  (Futures, continuations, …)
World ported to Racket, Java, JavaScript…
;;; Simple world program. The world is a number.

;;; reset: world -> world
;;; Reset the world back to zero.
(define (reset w) 0)

;;; GUI: world -> gui
(define (draw-gui w)
  (list (js-button reset '')
        (list (js-text (number->string w)))))

(big-bang 0 ;; initial world
  (on-tick add1 1)
  (on-draw draw-gui))
The World is Not Enough
Automatic adaptation of stateful APIs

In-place updates of world state

“Universe” for distributed computing

Moore versus Mealy
Take-Away Messages

• Programming with functions creates a virtuous cycle

• Middle-schoolers can write tests, survive code-reviews, etc.

• Programming models matter, especially across platforms
www.bootstrapworld.org

www.facebook.com/BootstrapWorld

Thanks:

Danny Yoo  Emmanuel Schanzer
Kathi Fisler  Matthias Felleisen
Zhe Zhang  (rest  🏀)