Functional Reactive Programming

Another approach to asynchronous systems

Federico Builes
federico@mheroin.com

Framework funcional declarativo para el procesamiento streams de eventos

```
$.ajax({
  url: apiUrl,
  method: apiMethod,
  data: { artist: artist, title: title },
  beforeSend: loader.start,
  success: function(data, status, jqXhr){
    loader.stop();
    showLyrics();
    $("#set-video").show();
    if (data === "Sorry, We don't have lyrics for this song yet.") {
      activateStep("step2-nolyrics");
    } else {
      $("#fetch-lyrics").hide();
      activateStep("step2");
    }

    write(data);
  },
  error: function(xhr, status, error){
    loader.stop();
    showLyrics();

    if (artist === "" || title === "") {
      write("You need to enter the artist's name and the song title.")
    } else {
      write("There has been an error processing the data. Please try again.");
    }
  }
});
```

```
$.ajax({
  url: apiUrl,
  method: apiMethod,
  data: { artist: artist, title: title },
  beforeSend: loader.start,
  success: function(data, status, jqXhr){
    loader.stop();
    showLyrics();
    $("#set-video").show();
    if (data === "Sorry, We don't have lyrics for this song yet.") {
      activateStep("step2-nolyrics");
    } else {
      $("#fetch-lyrics").hide();
      activateStep("step2");
    }

    write(data);
  },
  error: function(xhr, status, error){
    loader.stop();
    showLyrics();

    if (artist === "" || title === "") {
      write("You need to enter the artist's name and the song title.")
    } else {
      write("There has been an error processing the data. Please try again.");
    }
  }
});
```

```
$.ajax({
  url: apiUrl,
  method: apiMethod,
  data: { artist: artist, title: title },
  beforeSend: loader.start,
  success: function(data, status, jqXhr){
    loader.stop();
    showLyrics();
    $("#set-video").show();
    if (data === "Sorry, We don't have lyrics for this song yet.") {
      activateStep("step2-nolyrics");
    } else {
      $("#fetch-lyrics").hide();
      activateStep("step2");
    }

    write(data);
  },
  error: function(xhr, status, error){
    loader.stop();
    showLyrics();

    if (artist === "" || title === "") {
      write("You need to enter the artist's name and the song title.")
    } else {
      write("There has been an error processing the data. Please try again.");
    }
  }
});
```

```
$.ajax({
      url: apiUrl,
      method: apiMethod,
      data: { artist: artist, title: title },
      beforeSend: loader.start,
      success: ajaxSuccess,
      error: ajaxError,
    });
```

```
$.ajax({
      url: apiUrl,
      method: apiMethod,
      data: { artist: artist, title: title },
      beforeSend: loader.start,
      success: ajaxSuccess,
      error: ajaxError,
    });
```

Continuation-passing Style

(CPS)

Continuation-passing style (CPS) is a style of programming in which **control is passed explicitly in the form of a continuation.**

A continuation is a data structure that **represents the computational process at a given point** in the process' execution.

```
function id(x) {
  return x;
}

// CPS

function id(x, cc) {
  cc(x);
}
```

```
function fact(n) {
  if (n == 0)
    return 1;
  else
    return n * fact(n-1);
}

// En CPS
function fact(n, cc) {
  if (n == 0)
    cc(1);
  else {
    fact(n - 1, function (val) {
      cc(n * val)
    });
  }
}
```

```
fact (5, function (n) {
  console.log(n) ;
})
```

```
(defun pyth (x y)
  (sqrt (+ (* x x) (* y y))))
```

```
(defun cps+ (x y cc)
  (funcall cc (+ x y)))

(defun cps* (x y cc)
  (funcall cc (* x y)))

(defun sqrt*(x cc)
  (funcall cc (sqrt x)))

(defun cps-pyth (x y cc)
  (cps* x x (lambda (x2)
    (cps* y y (lambda (y2)
      (cps+ x2 y2 (lambda (sum)
        (sqrt* sum cc))))))))
```

viva Lisp =)

```haskell
add_cps :: Int -> Int -> (Int -> r) -> r
add_cps x y k = k (x + y)

square_cps :: Int -> (Int -> r) -> r
square_cps x k = k (x * x)

pythagoras_cps :: Int -> Int -> (Int -> r) -> r
pythagoras_cps x y k =
 square_cps x $ \x_squared ->
 square_cps y $ \y_squared ->
 add_cps x_squared y_squared $ \sum_of_squares ->
 k sum_of_squares
```

```haskell
add_cps :: Int -> Int -> (Int -> r) -> r
add_cps x y k = k (x + y)

square_cps :: Int -> (Int -> r) -> r
square_cps x k = k (x * x)

pythagoras_cps :: Int -> Int -> (Int -> r) -> r
pythagoras_cps x y k =
 square_cps x $ \x_squared ->
 square_cps y $ \y_squared ->
 add_cps x_squared y_squared $ \sum_of_squares ->
 k sum_of_squares
```

Compiladores/Analizadores

Cambios en Flujo

Concurrencia

Sistemas Distribuidos

Sistemas con Latencia Inherente

Callback Hell

```
$.ajax({
  url: apiUrl,
  method: apiMethod,
  data: { artist: artist, title: title },
  beforeSend: loader.start,
  success: function(data, status, jqXhr){
    loader.stop();
    showLyrics();
    $("#set-video").show();
    if (data === "Sorry, We don't have lyrics for this song yet.") {
      activateStep("step2-nolyrics");
    } else {
      $("#fetch-lyrics").hide();
      activateStep("step2");
    }

    write(data);
  },
  error: function(xhr, status, error){
    loader.stop();
    showLyrics();

    if (artist === "" || title === "") {
      write("You need to enter the artist's name and the song title.")
    } else {
      write("There has been an error processing the data. Please try again.");
    }
  }
});
```

```
$.ajax({
      url: apiUrl,
      method: apiMethod,
      data: { artist: artist, title: title },
      beforeSend: loader.start,
      success: ajaxSuccess,
      error: ajaxError,
    });
```

```
$.ajax({
      url: apiUrl,
      method: apiMethod,
      data: { artist: artist, title: title },
      beforeSend: loader.start,
      success: ajaxSuccess,
      error: ajaxError,
    });
```

```
$ find . -name *.js | xargs cat | wc -l
 1019

$ ack ”\.ajax\({” | wc -l
  48
```

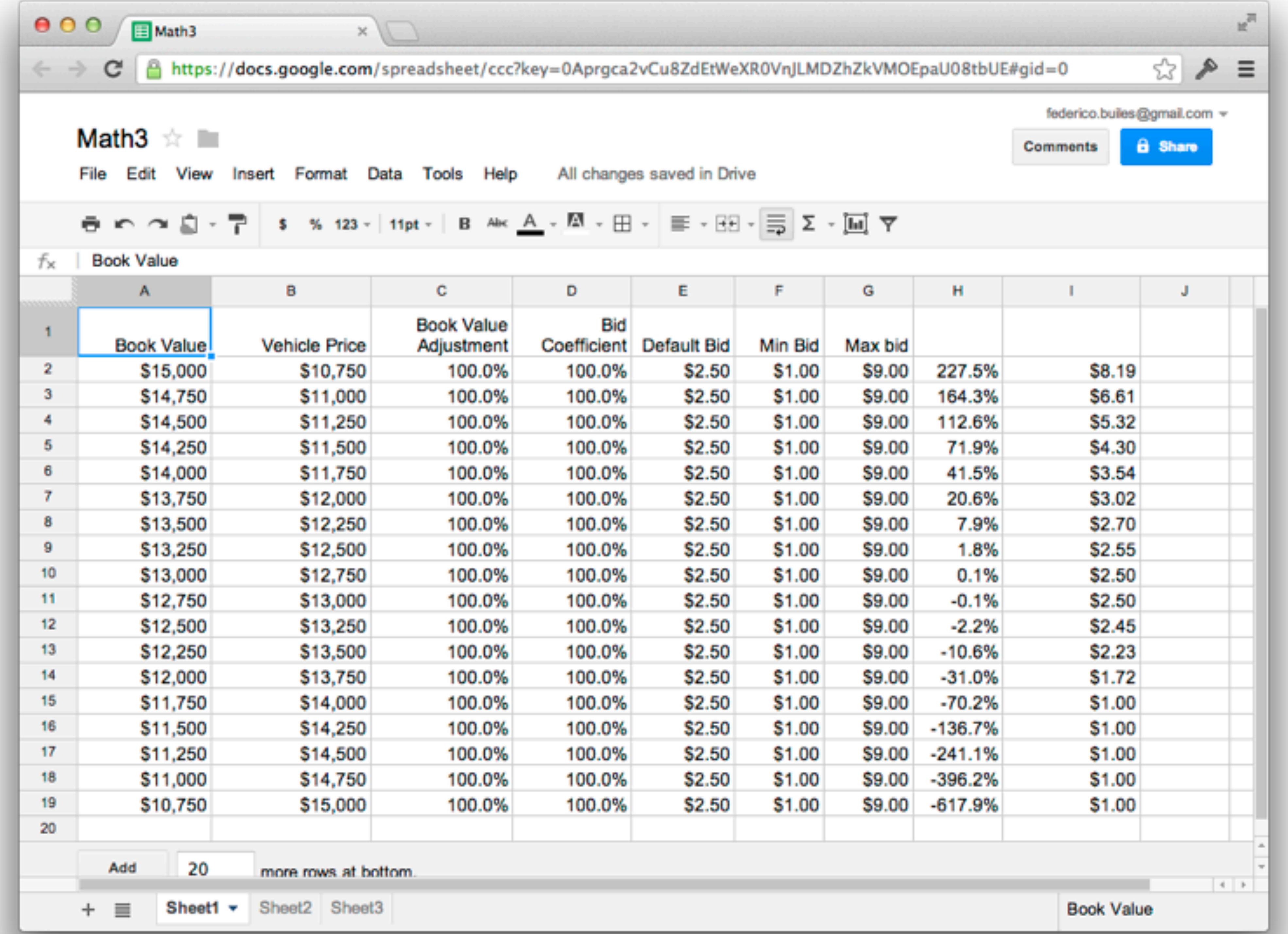
goto : structured programming

::

callbacks : async programming

Functional Reactive Programming

Functional Reactive Programming


Math3
https://docs.google.com/spreadsheet/ccc?key=0Aprgca2vCu8ZdEtWeXR0VnJLMDZhZkVMOEpaU08tbUE#gid=0
federico.builes@gmail.com
Math3
Comments
Share
File
Edit
View
Insert
Format
Data
Tools
Help
All changes saved in Drive
$
%
123
11pt
B
Abc
A
Σ
fx
Book Value
Add
20
more rows at bottom.
+
Sheet1
Sheet2
Sheet3
Book Value

	A	B	C	D	E	F	G	H	I	J
1	Book Value	Vehicle Price	Book Value Adjustment	Bid Coefficient	Default Bid	Min Bid	Max bid			
2	$15,000	$10,750	100.0%	100.0%	$2.50	$1.00	$9.00	227.5%	$8.19	
3	$14,750	$11,000	100.0%	100.0%	$2.50	$1.00	$9.00	164.3%	$6.61	
4	$14,500	$11,250	100.0%	100.0%	$2.50	$1.00	$9.00	112.6%	$5.32	
5	$14,250	$11,500	100.0%	100.0%	$2.50	$1.00	$9.00	71.9%	$4.30	
6	$14,000	$11,750	100.0%	100.0%	$2.50	$1.00	$9.00	41.5%	$3.54	
7	$13,750	$12,000	100.0%	100.0%	$2.50	$1.00	$9.00	20.6%	$3.02	
8	$13,500	$12,250	100.0%	100.0%	$2.50	$1.00	$9.00	7.9%	$2.70	
9	$13,250	$12,500	100.0%	100.0%	$2.50	$1.00	$9.00	1.8%	$2.55	
10	$13,000	$12,750	100.0%	100.0%	$2.50	$1.00	$9.00	0.1%	$2.50	
11	$12,750	$13,000	100.0%	100.0%	$2.50	$1.00	$9.00	-0.1%	$2.50	
12	$12,500	$13,250	100.0%	100.0%	$2.50	$1.00	$9.00	-2.2%	$2.45	
13	$12,250	$13,500	100.0%	100.0%	$2.50	$1.00	$9.00	-10.6%	$2.23	
14	$12,000	$13,750	100.0%	100.0%	$2.50	$1.00	$9.00	-31.0%	$1.72	
15	$11,750	$14,000	100.0%	100.0%	$2.50	$1.00	$9.00	-70.2%	$1.00	
16	$11,500	$14,250	100.0%	100.0%	$2.50	$1.00	$9.00	-136.7%	$1.00	
17	$11,250	$14,500	100.0%	100.0%	$2.50	$1.00	$9.00	-241.1%	$1.00	
18	$11,000	$14,750	100.0%	100.0%	$2.50	$1.00	$9.00	-396.2%	$1.00	
19	$10,750	$15,000	100.0%	100.0%	$2.50	$1.00	$9.00	-617.9%	$1.00	
20										

Functional

Reactive

Programming

Behaviors

(signals)

Valores reactivos que varían en el tiempo

```
Behavior a = Time → a
```

El valor del behavior ***s*** en el tiempo ***t*** es ***s(t)***

Event

Sencuencias de ocurrencias de eventos en el tiempo

```
Event a = Time × a
```

Pares: (tiempo, información sobre la ocurrencia evento)

```
leftClick :: Event ()
 keyPress :: Event Char
```

Behaviors Constantes

```
red :: Behavior Color
  1 :: Behavior Real
```

Behaviors Variables

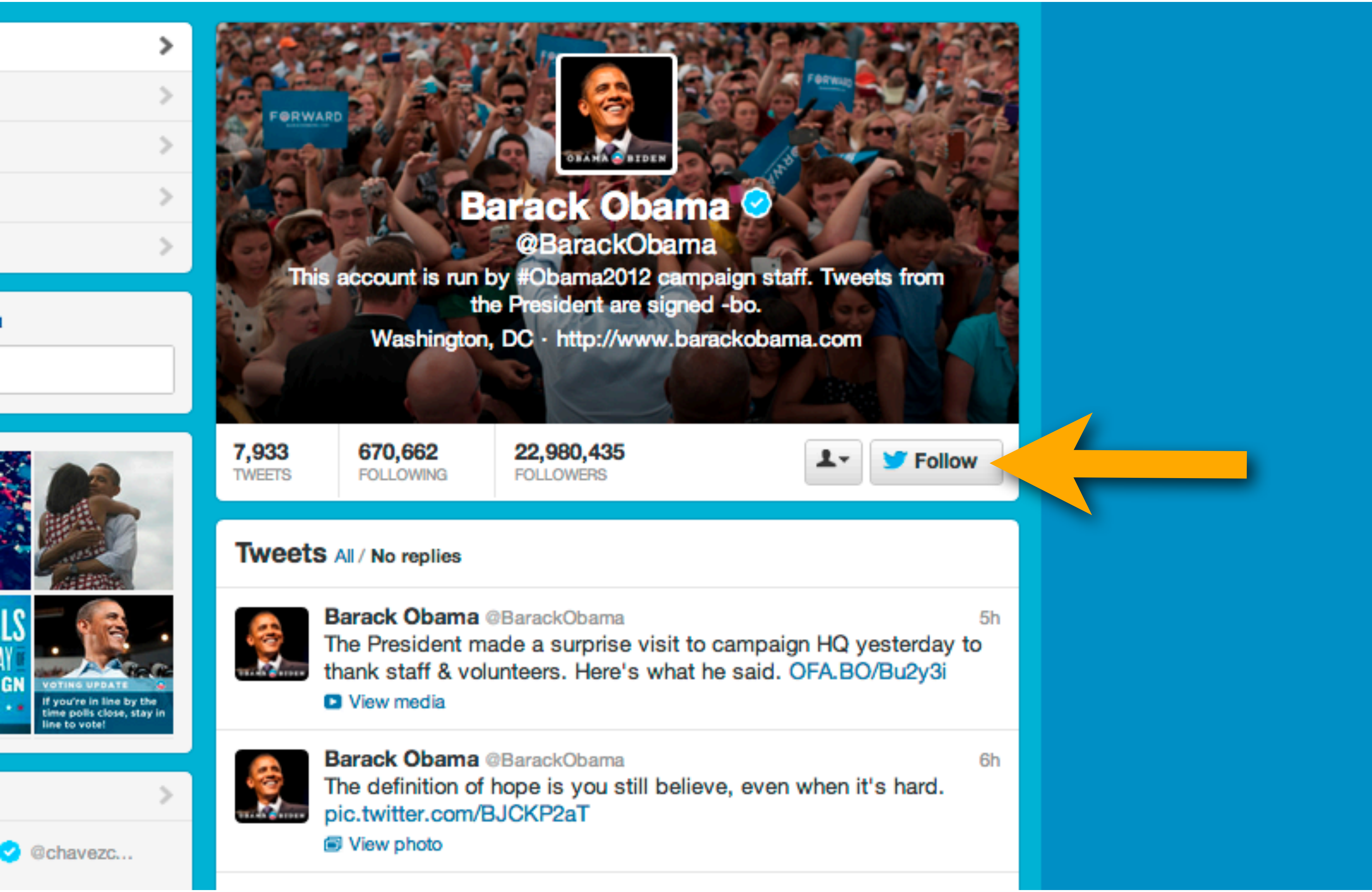
```
time :: Behavior Time*
```

```
-=> :: Eventα -> β -> Eventβ

color :: Behavior Color
color = red `until` (leftClick -=> blue)

circ :: Behavior Region
circ = translate (cos time, sin time) (circle 1)

ball :: Behavior Picture
ball = paint color circ
```



FORWARD
OBAMA BIDEN
Follow
VOTING UPDATE
If you're in line by the time polls close, stay in line to vote!
@chavezc...

Barack Obama

@BarackObama

This account is run by #Obama2012 campaign staff. Tweets from the President are signed -bo.

Washington, DC · http://www.barackobama.com

7,933 TWEETS

670,662 FOLLOWING

22,980,435 FOLLOWERS

Tweets All / No replies

Barack Obama @BarackObama 5h

The President made a surprise visit to campaign HQ yesterday to thank staff & volunteers. Here's what he said. OFA.BO/Bu2y3i

View media

Barack Obama @BarackObama 6h

The definition of hope is you still believe, even when it's hard. pic.twitter.com/BJCKP2aT

View photo

```ruby
def follow
  res = write_to_db
  res = notify_user(res)
  return update_ui(res)
end

def write_to_db
  # ...
end

def notify_user
  # ...
end

def update_ui
  # ...
end
```

blocking

```
def follow
  write_to_db.callback do |a|
    notify_user(a).callback do |b|
      update_ui(b).callback do |resp|
        puts resp
      end
    end
  end
end
```

async

```
# follow es el evento
def follow
  tie(:write_to_db, :notify_user, :update_ui)
end
```

reactive

Referencias

- Gerald Jay Sussman and Guy L. Steele, Jr. "Scheme: An interpreter for extended lambda calculus". AI Memo 349: 19, December 1975.

- Edsger W. Dijkstra. Go To Statement Considered Harmful. Communications of the ACM, Vol. 11, No. 3, March 1968, pp. 147-148.

- Conal Elliott and Paul Hudak. "Functional Reactive Animation". ICFP 1997. http://conal.net/papers/icfp97/.

- Zhanyong Wan and Paul Hudak. "Functional Reactive Programming from First Principles". In ACM SIGPLAN Conference on Programming Language Design and Implementation, pp. 242-252, 2000.

- Conal Elliott, "Push-Pull Functional Reactive Programming". Haskell Symposium, 2009. http://conal.net/papers/push-pull-frp/push-pull-frp.pdf. Video: http://www.vimeo.com/6686570

- http://elm-lang.org/learn/What-is-FRP.elm

- http://en.wikibooks.org/wiki/Haskell/Continuation_passing_style