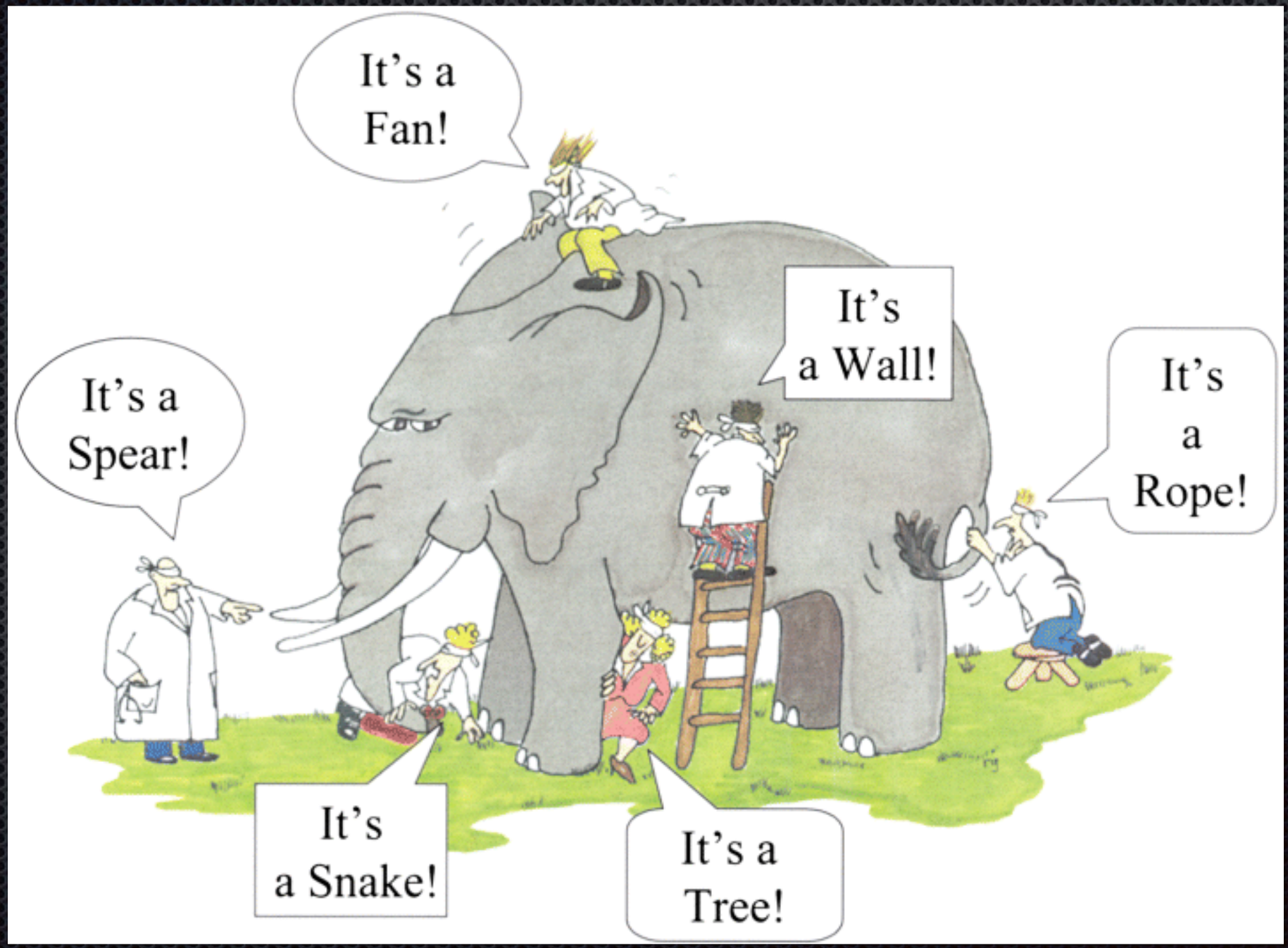
Reactive Programming Dataflow Constraints

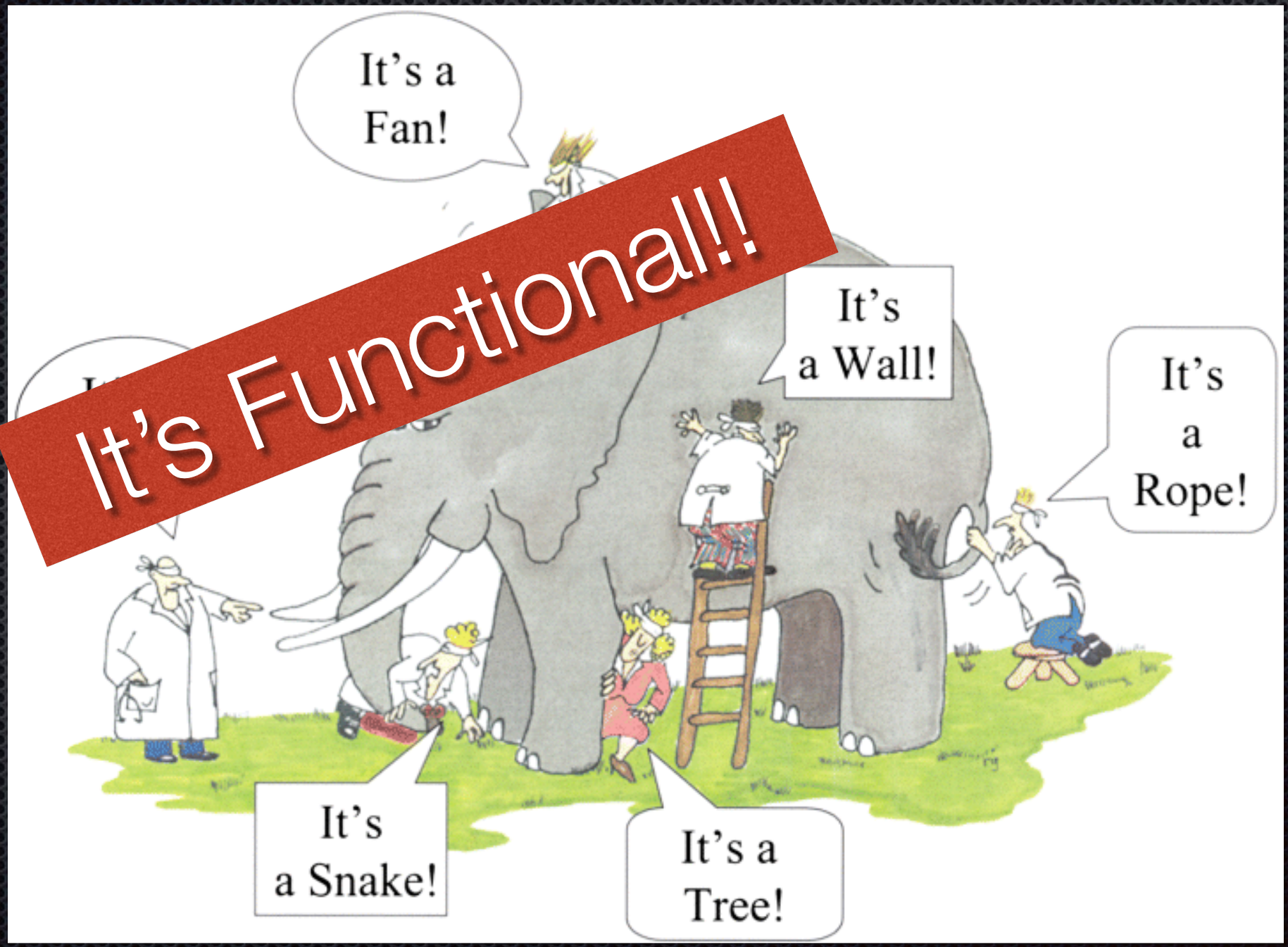
Marcel Weiher

@mpweiher

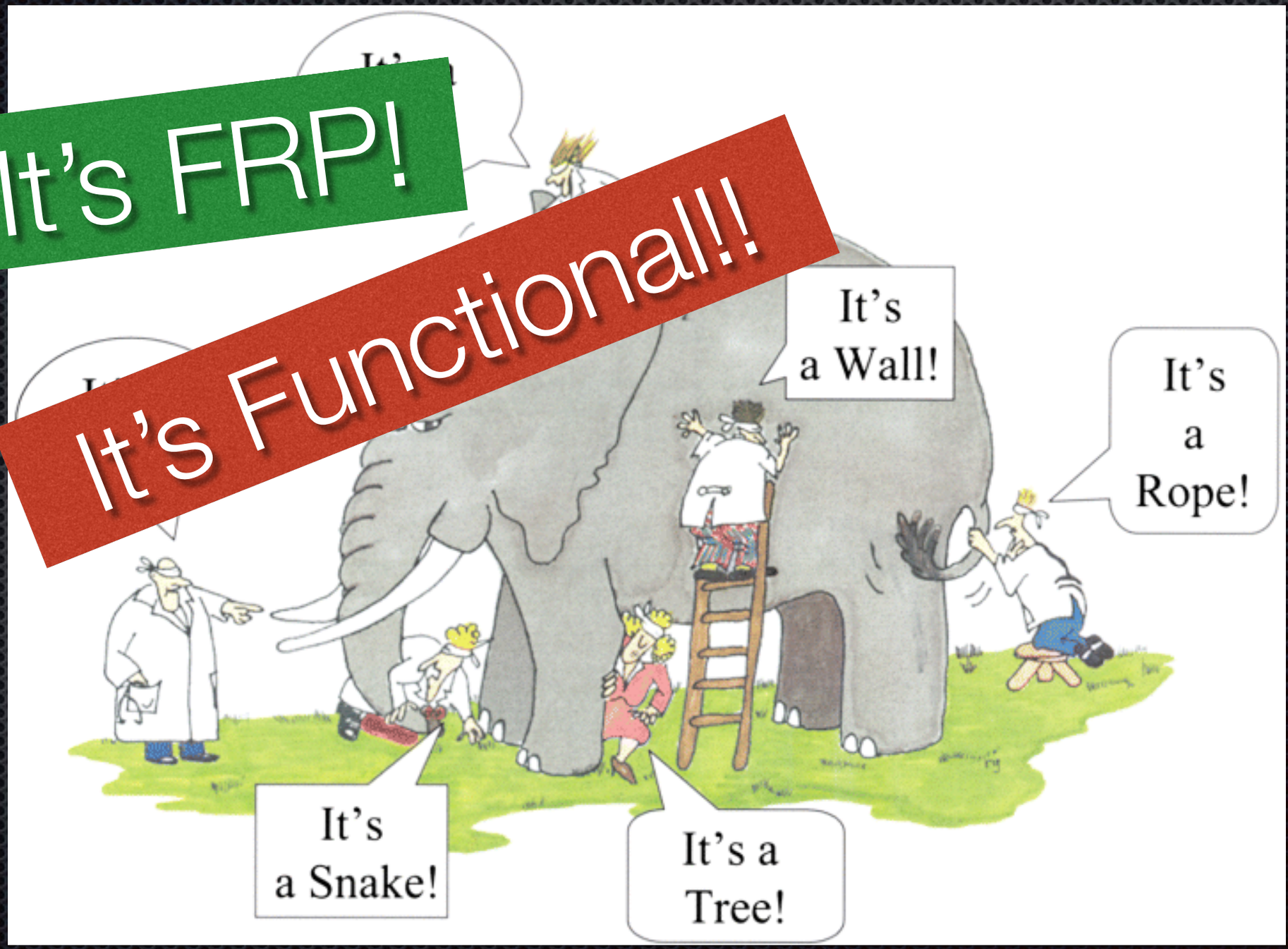
What is Reactive Programming?



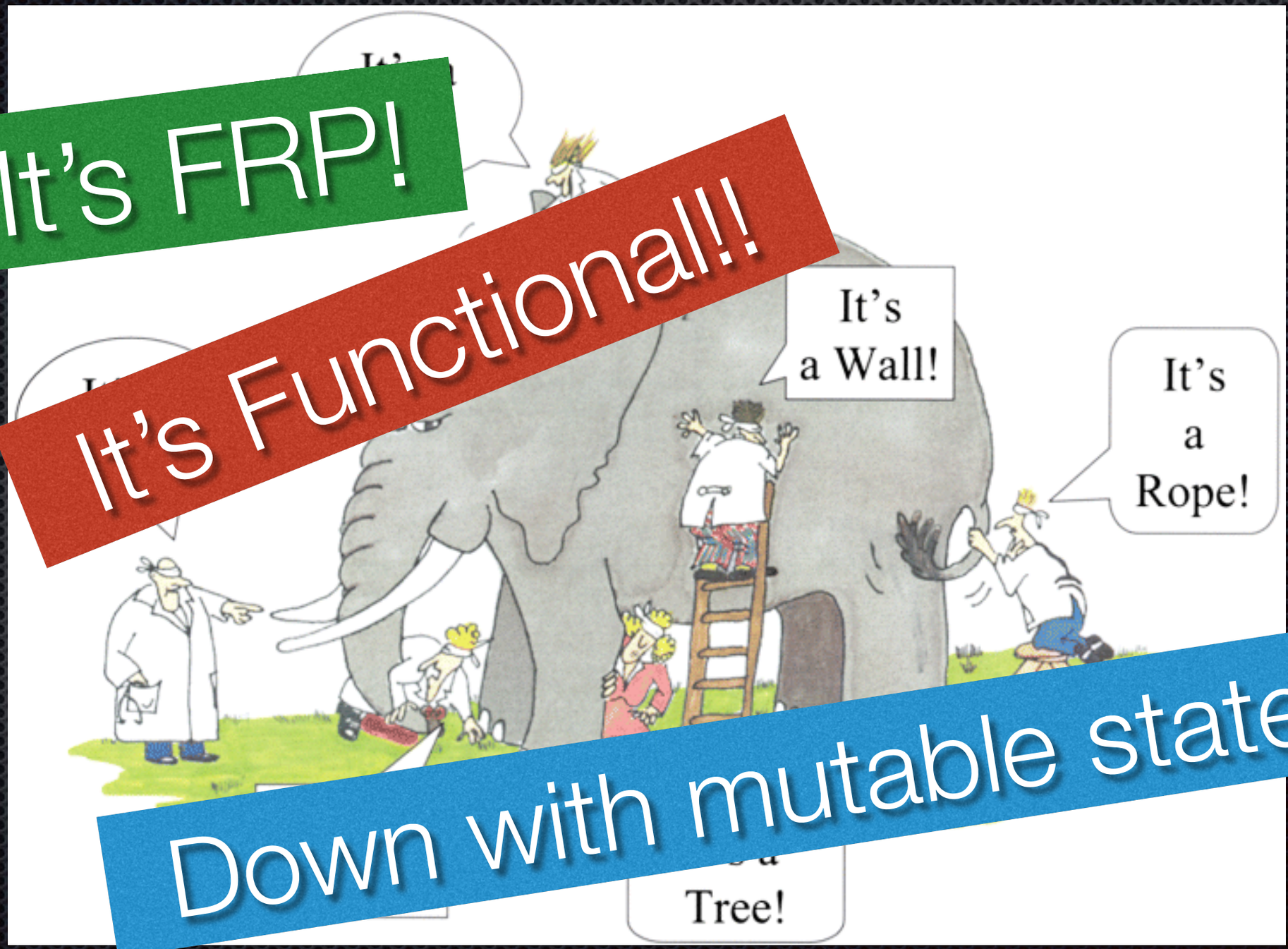
What is Reactive Programming?



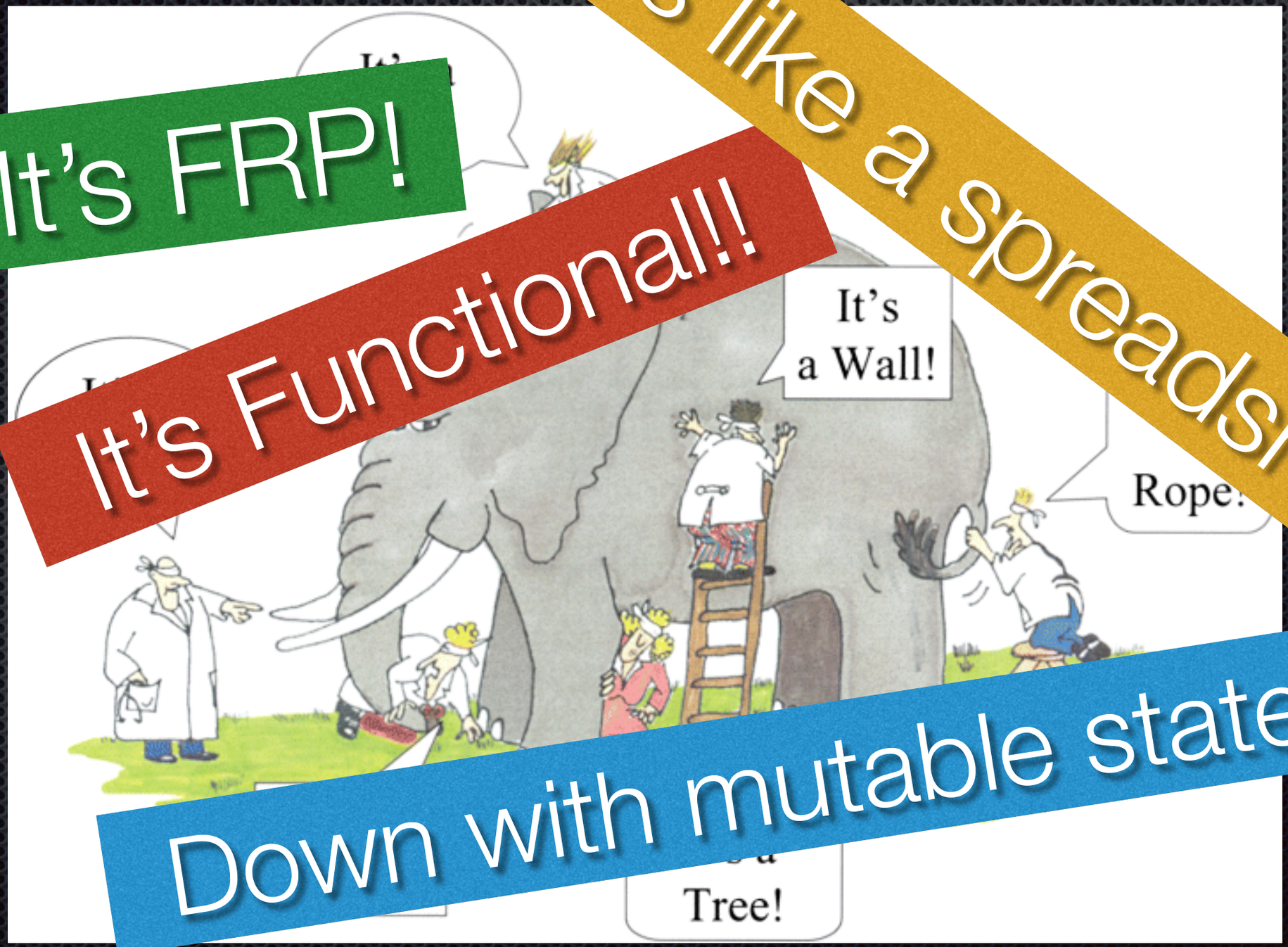
What is Reactive Programming?



What is Reactive Programming?



What is Reactive Programmin



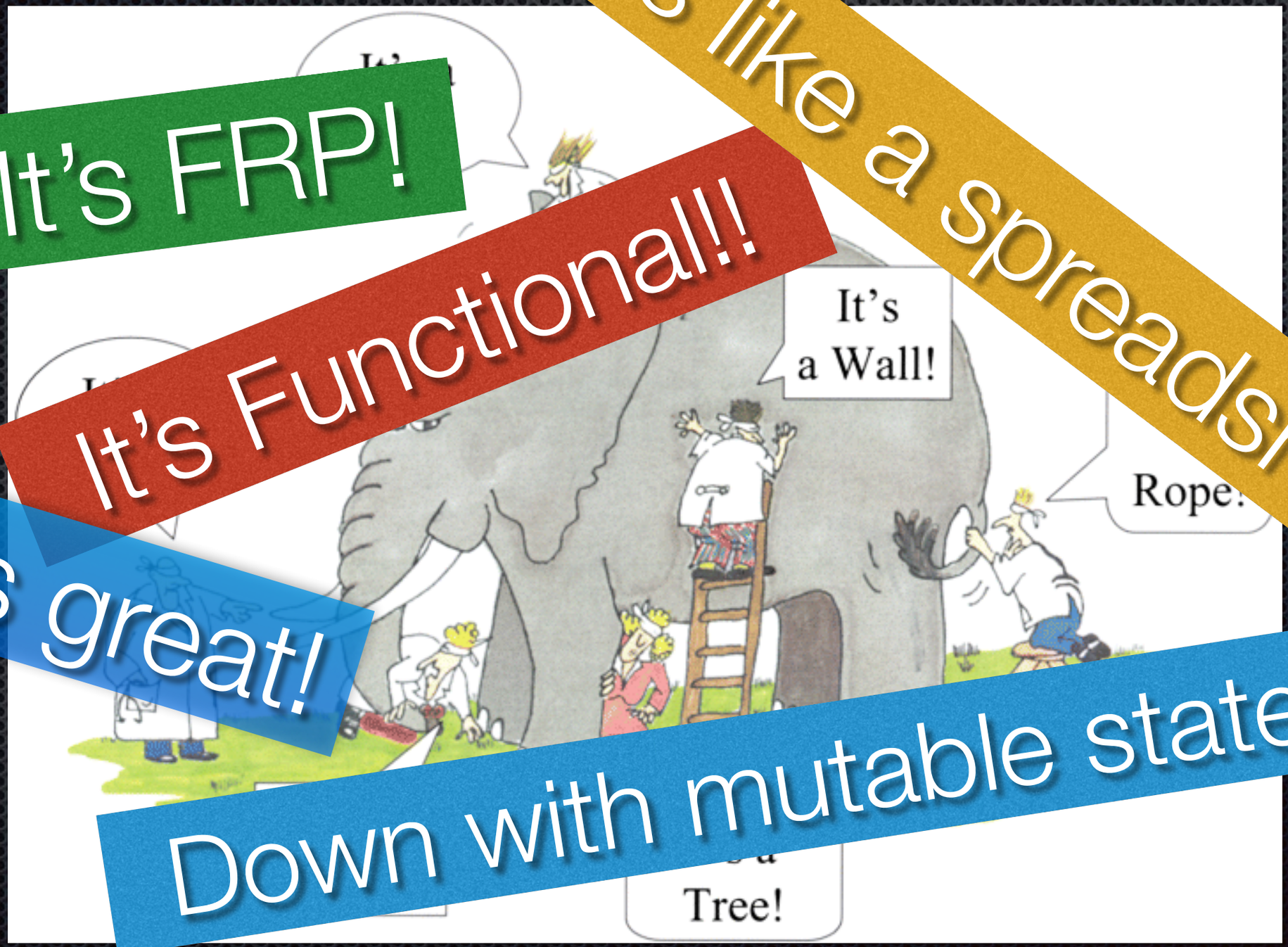
It's FRP!

It's Functional!!

It's like a spreadsheet

Down with mutable state!!

What is Reactive Programmin



It's FRP!

It's Functional!!

It's like a spreadsheet

It's great!

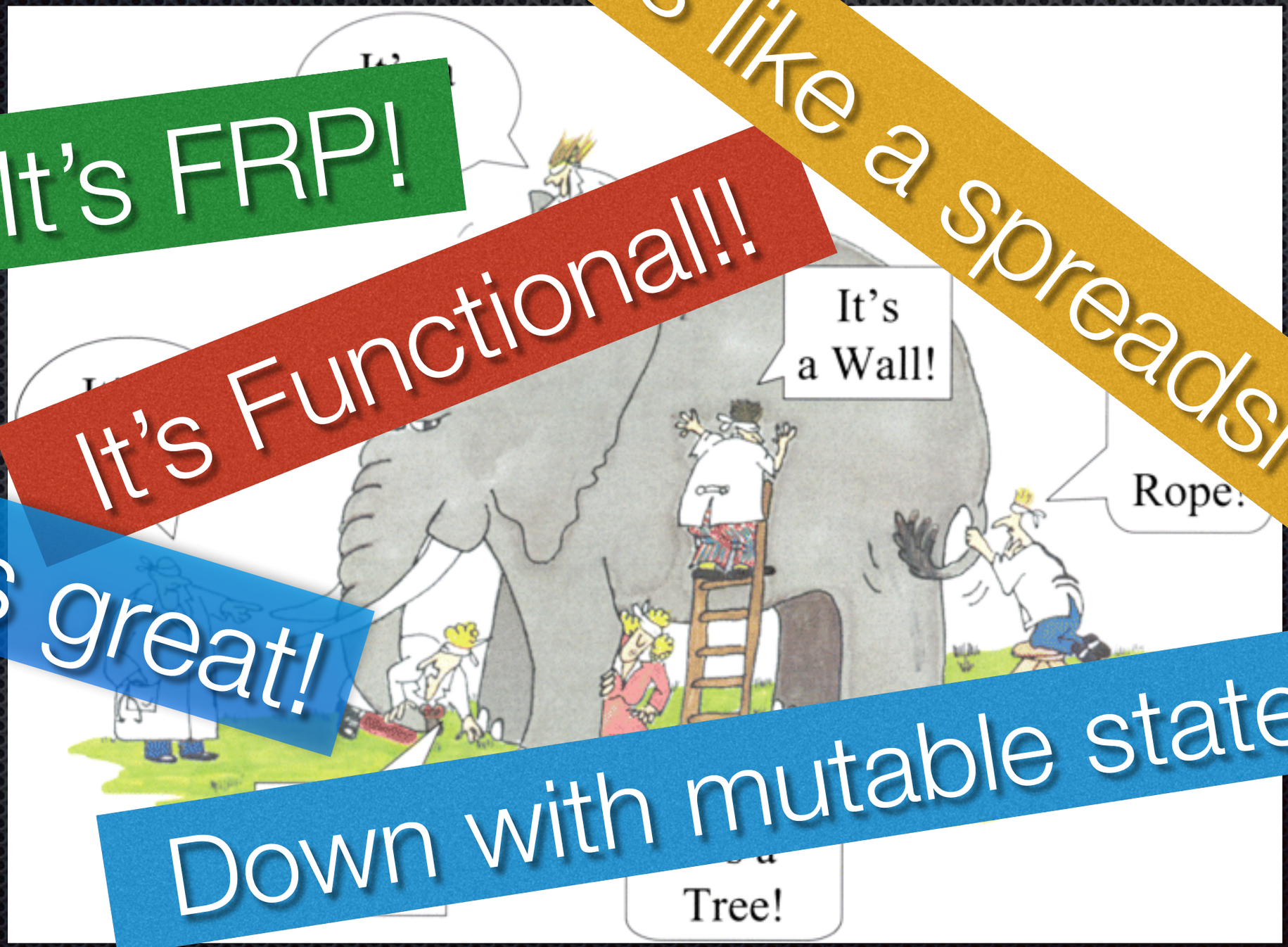
Down with mutable state!!

What is Reactive Programmin

What is Reactive Programming *not*?

- FRP was somewhat of a misnomer, better: “Functional Temporal Programming” — *Conal Eliot, creator of FRP*
- Rx is not functional and not reactive — *Erik Meijer, creator of Rx*
- RAC, Elm, Rx etc. have nothing to do with FRP — *Conal Eliot* ***and*** *Erik Meijer*

What is Reactive Programmin



It's FRP!

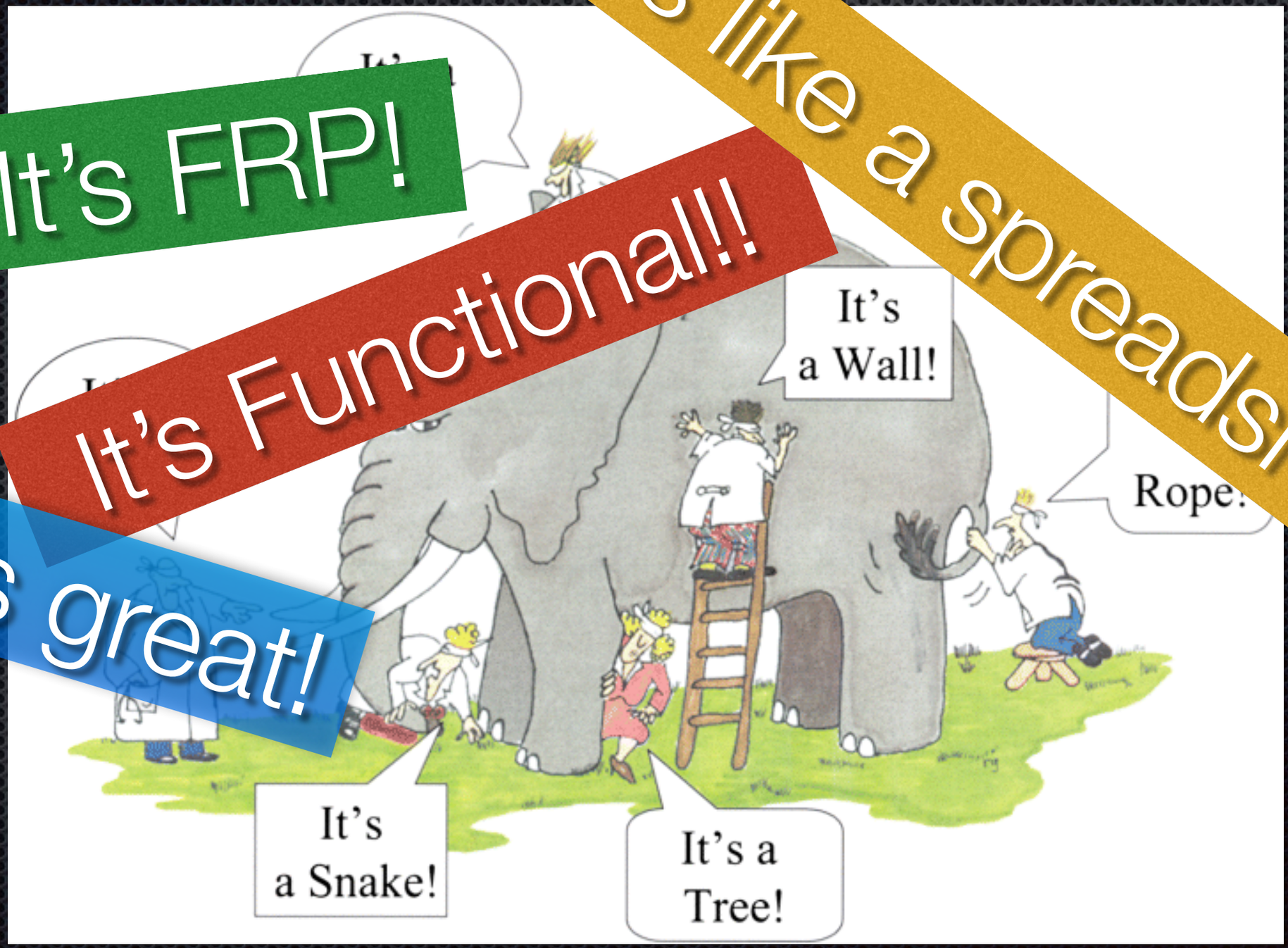
It's Functional!!

It's like a spreadsheet

It's great!

Down with mutable state!!

What is Reactive Programmin



It's FRP!

It's Functional!!

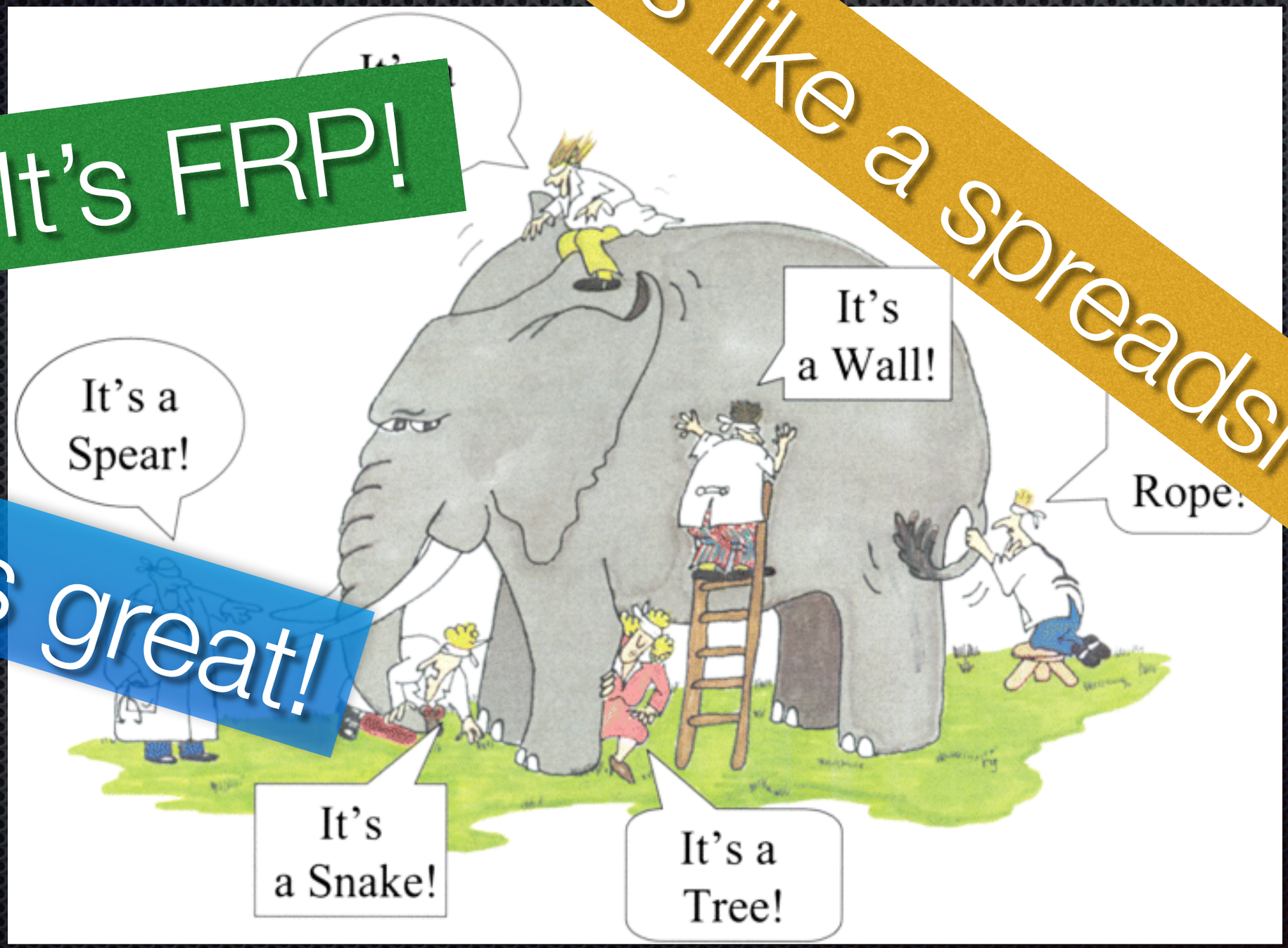
It's like a spreadsheet

It's great!

What is Reactive Programmin

It's FRP!

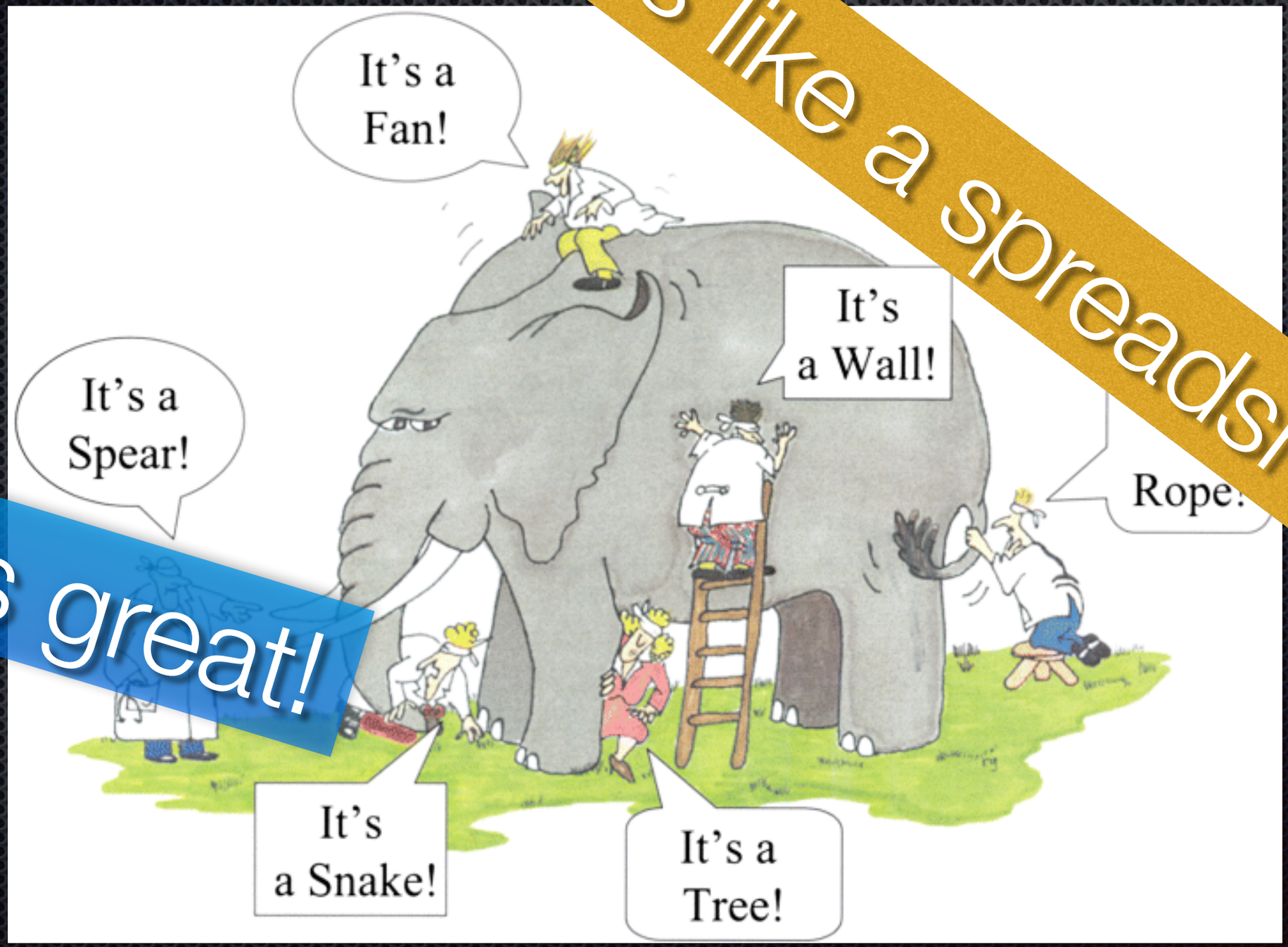
It's like a spreadsheet



It's great!

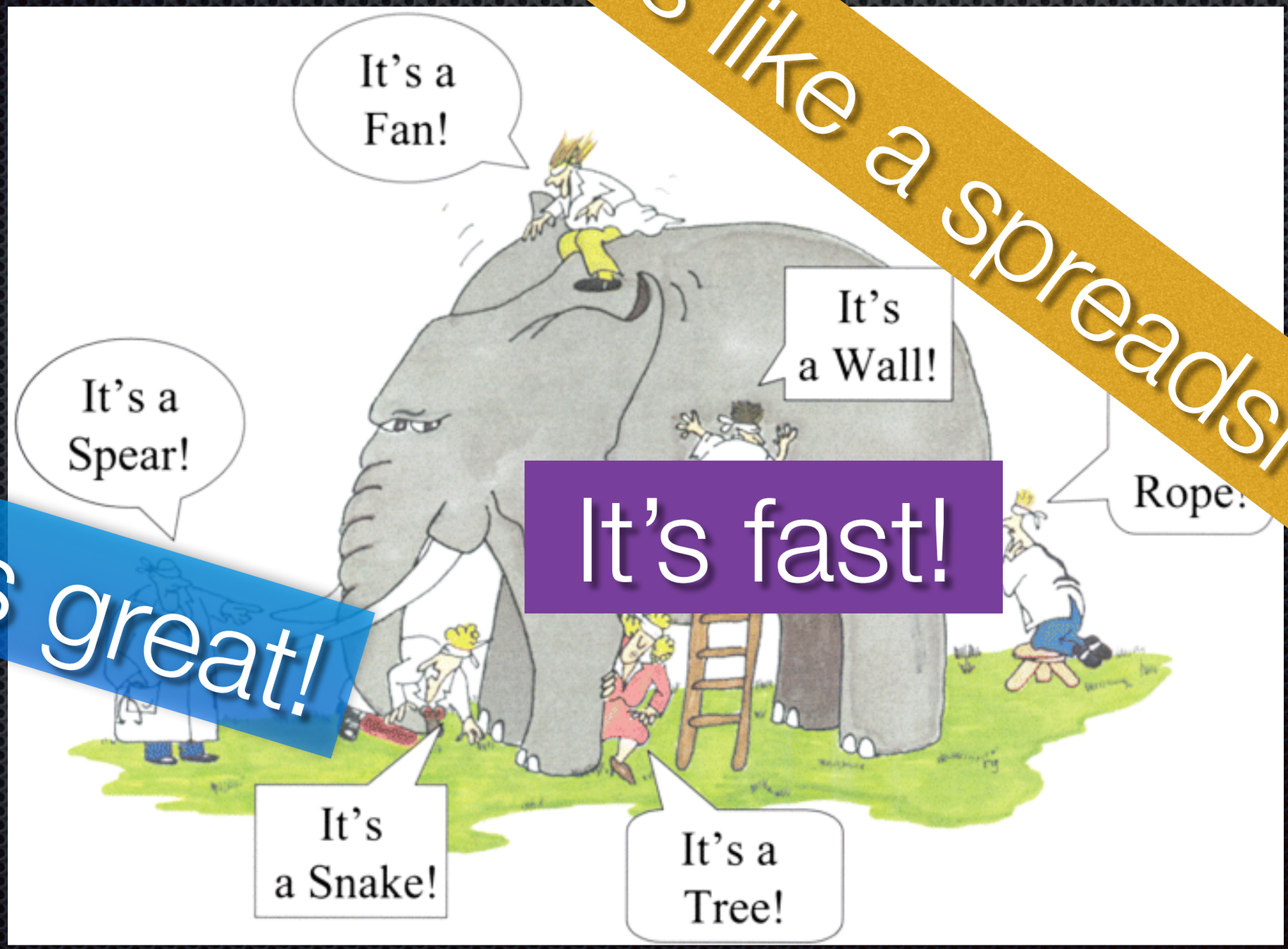
What is Reactive Programmin

It's like a spreadsheet



It's great!

What is Reactive Programmin



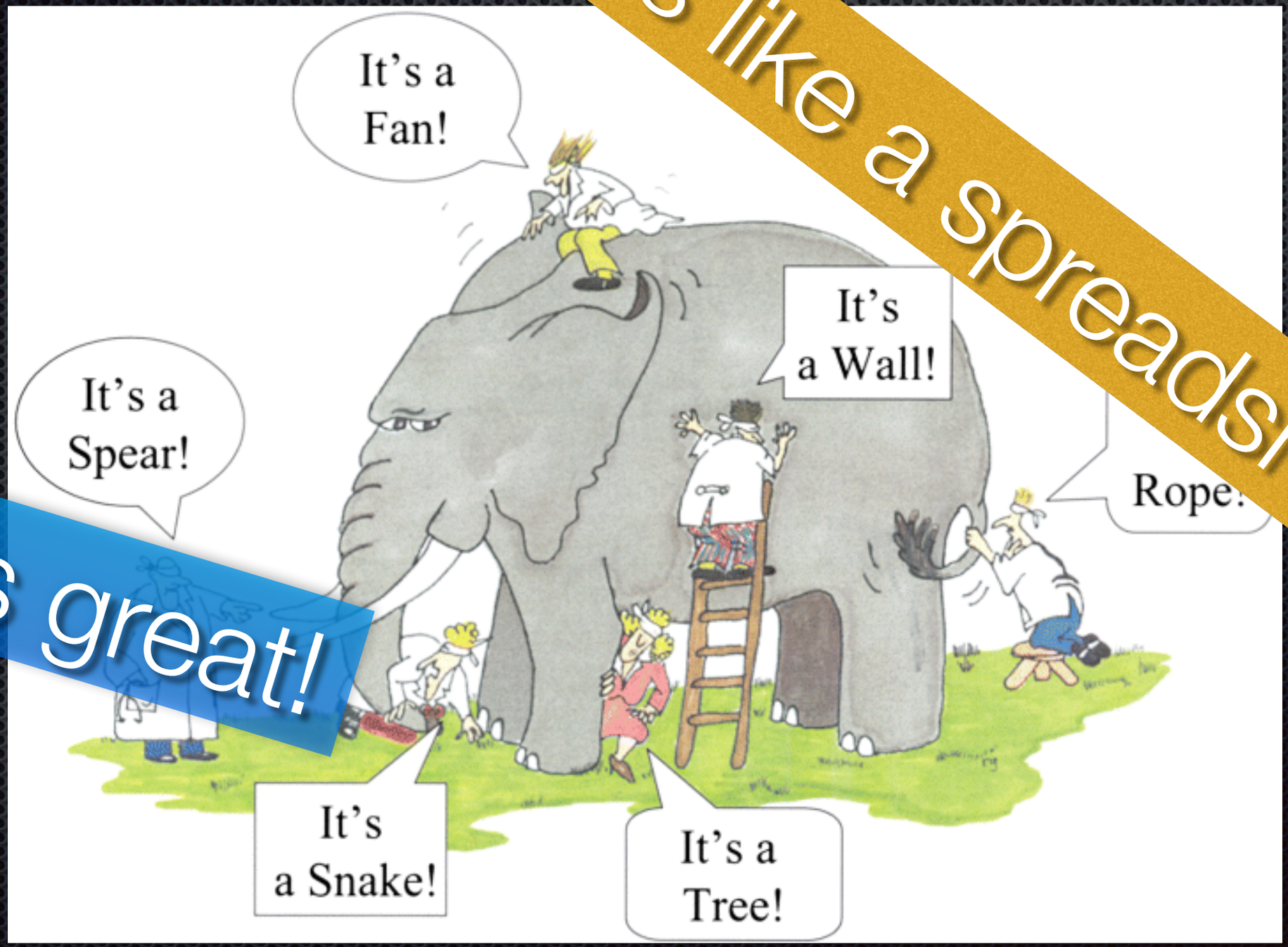
It's like a spreadsheet

It's great!

It's fast!

What is Reactive Programmin

It's like a spreadsheet



It's great!

What is Reactive Programming?

A Survey on Reactive Programming

ENGINEER BAINOMUGISHA, ANDONI LOMBIDE CARRETON, TOM VAN CUTSEM,
STIJN MOSTINCKX, and WOLFGANG DE MEUTER, Vrije Universiteit Brussel

ACM Computing Surveys, Vol. 45, No. 4, Article 52, Publication date: August 2013.

The reactive programming paradigm is based on the synchronous dataflow programming paradigm [Lee and Messerschmitt 1987] but with relaxed real-time constraints.

What is Reactive Programming?

Synchronous Data Flow

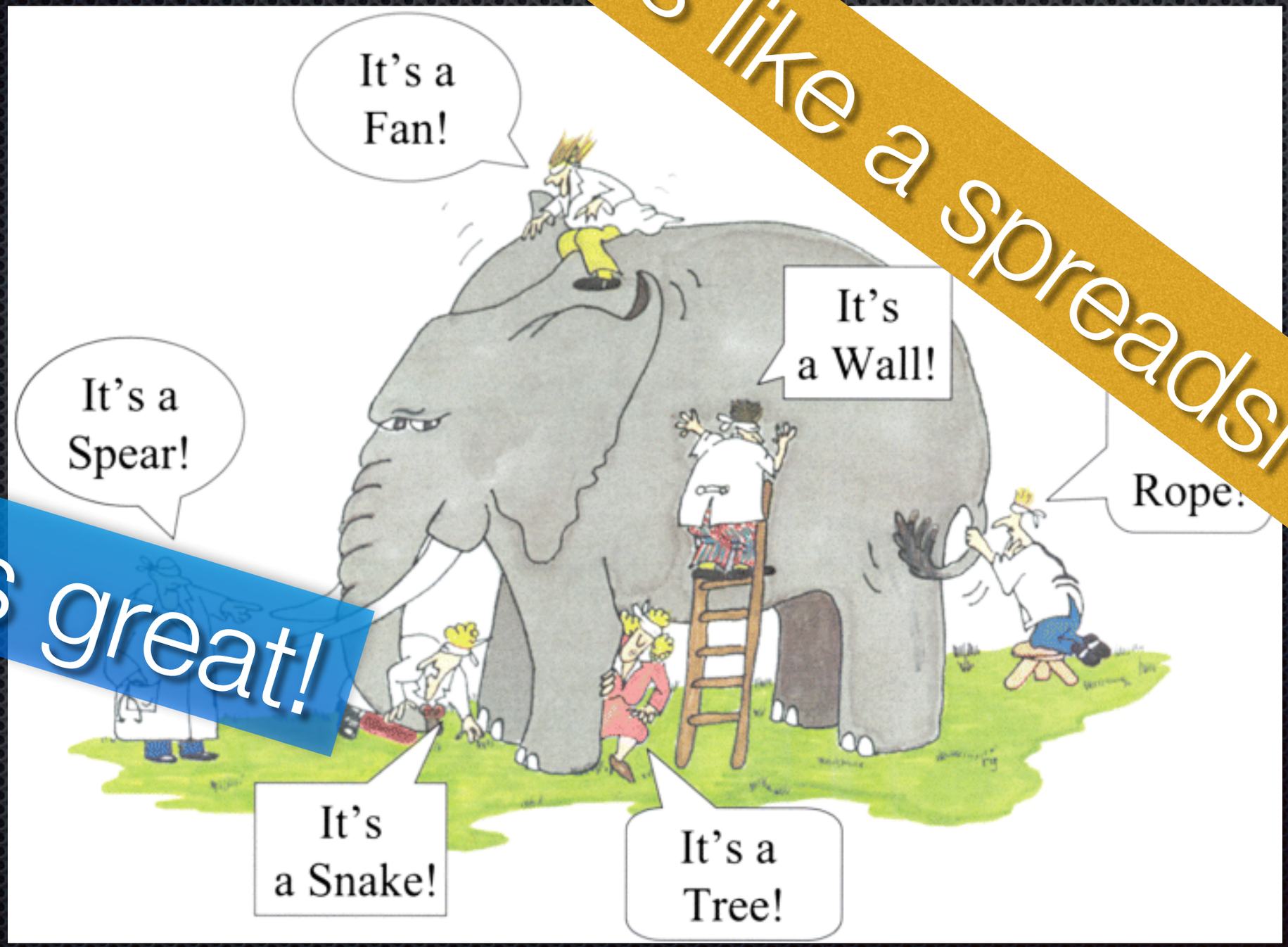
EDWARD A. LEE, MEMBER, IEEE, AND DAVID G. MESSERSCHMITT, FELLOW, IEEE

node represents a function and ...

Synchronous data flow (SDF) *is a special case of data flow (either atomic or large grain) in which the number of data samples produced or consumed by each node on each invocation is specified* a priori. *Nodes can be scheduled statically (at compile time) onto single or parallel programmable processors so the run-time overhead usually associated with data flow evaporates. Multiple sam-*

What is Reactive Programmin

It's like a spreadsheet

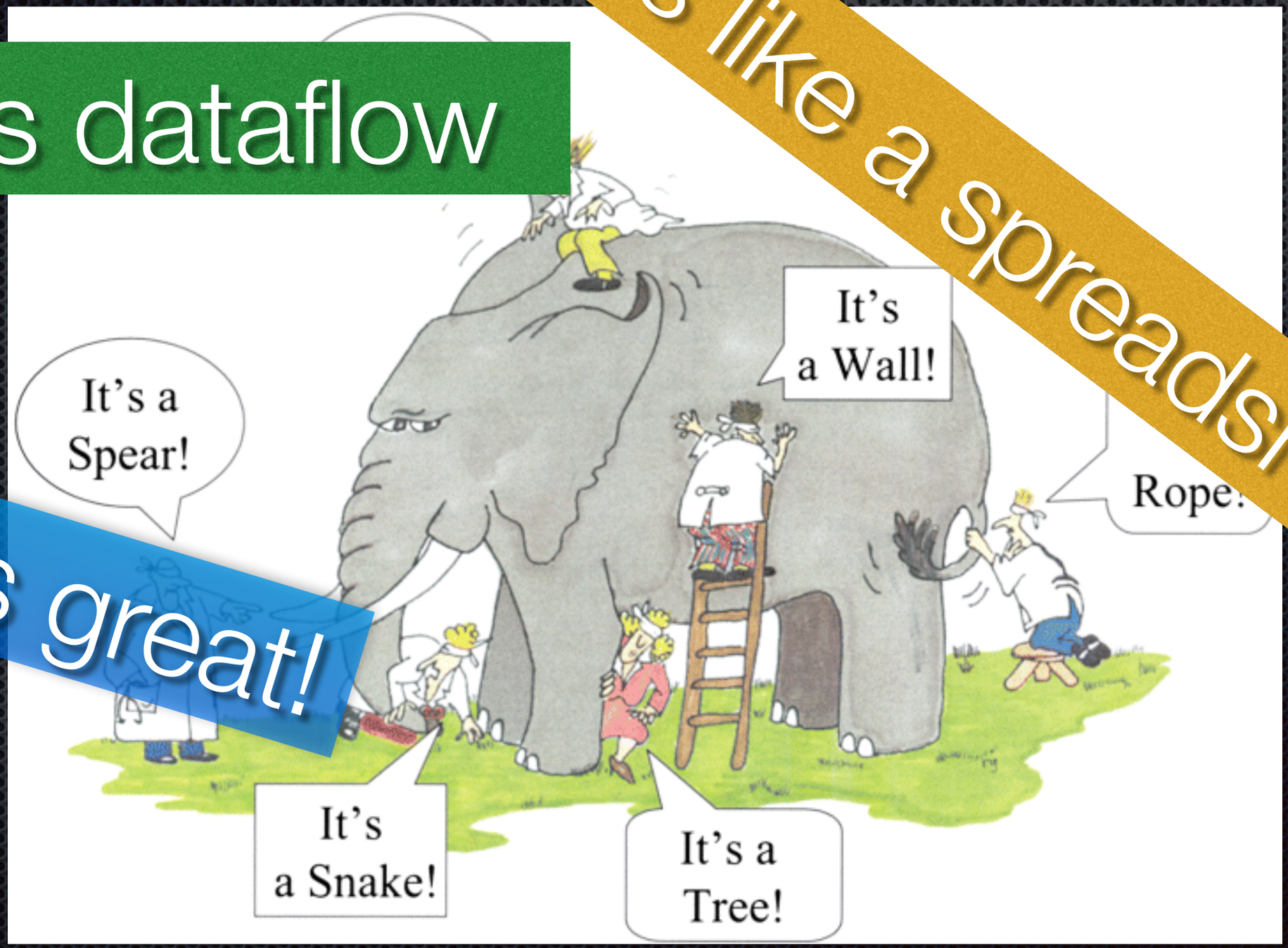


It's great!

What is Reactive Programmin

It's dataflow

It's like a spreadsheet



It's great!

What is Reactive Programming?

A. Reactive Systems

Reactive systems have been defined as computing systems which continuously interact with a given physical environment, when this environment is unable to synchronize logically with the system (for instance it cannot wait). Response times of the system must then meet requirements induced by the environment. This class of systems has been proposed [6], [21] so as to distinguish them from *transformational* systems—i.e., classical programs whose data are available at their beginning, and which provide results when terminating—and from *interactive systems*

What is Reactive Programming?

results when terminating—and from *interactive systems* which interact continuously with environments that possess synchronization capabilities (for instance operating systems). Reactive systems apply mainly to automatic process control and monitoring, and signal processing,—but also to systems such as communication protocols and man-machine interfaces when required response times are very small. Generally, these systems share some important features:

What is Reactive Programming?

The Synchronous Data Flow Programming Language LUSTRE

NICHOLAS HALBWACHS, PAUL CASPI, PASCAL RAYMOND, AND DANIEL PILAUD

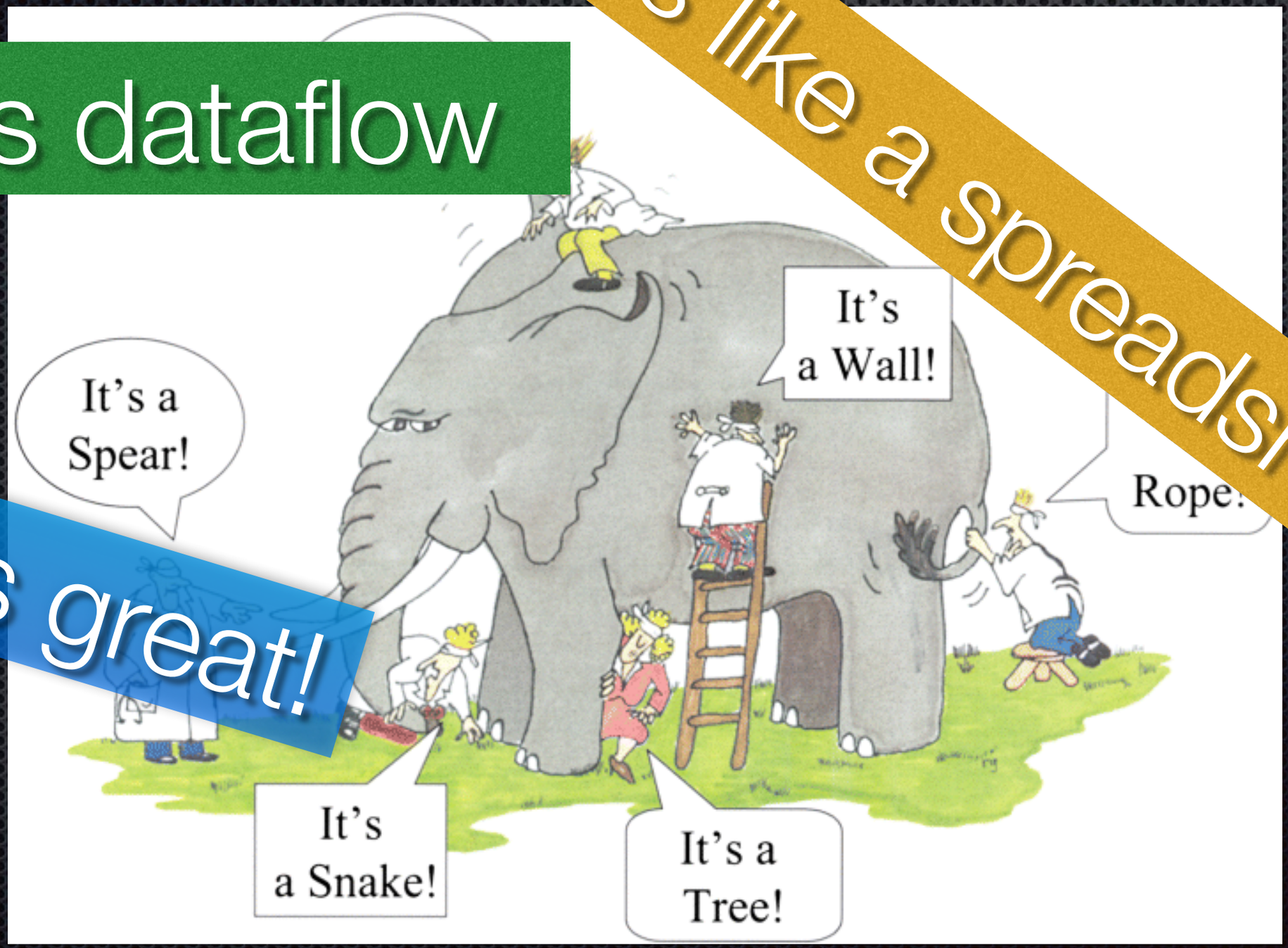
Invited Paper

Manuscript received September 20, 1990; revised March 2, 1991. This work was supported in part by the French Ministère de la Recherche within contract "Informatique 88," and in part by PRC C^3 (CNRS) IMAG/LGI-Grenoble VERILOG-Grenoble.

What is Reactive Programmin

It's dataflow

It's like a spreadsheet



It's great!

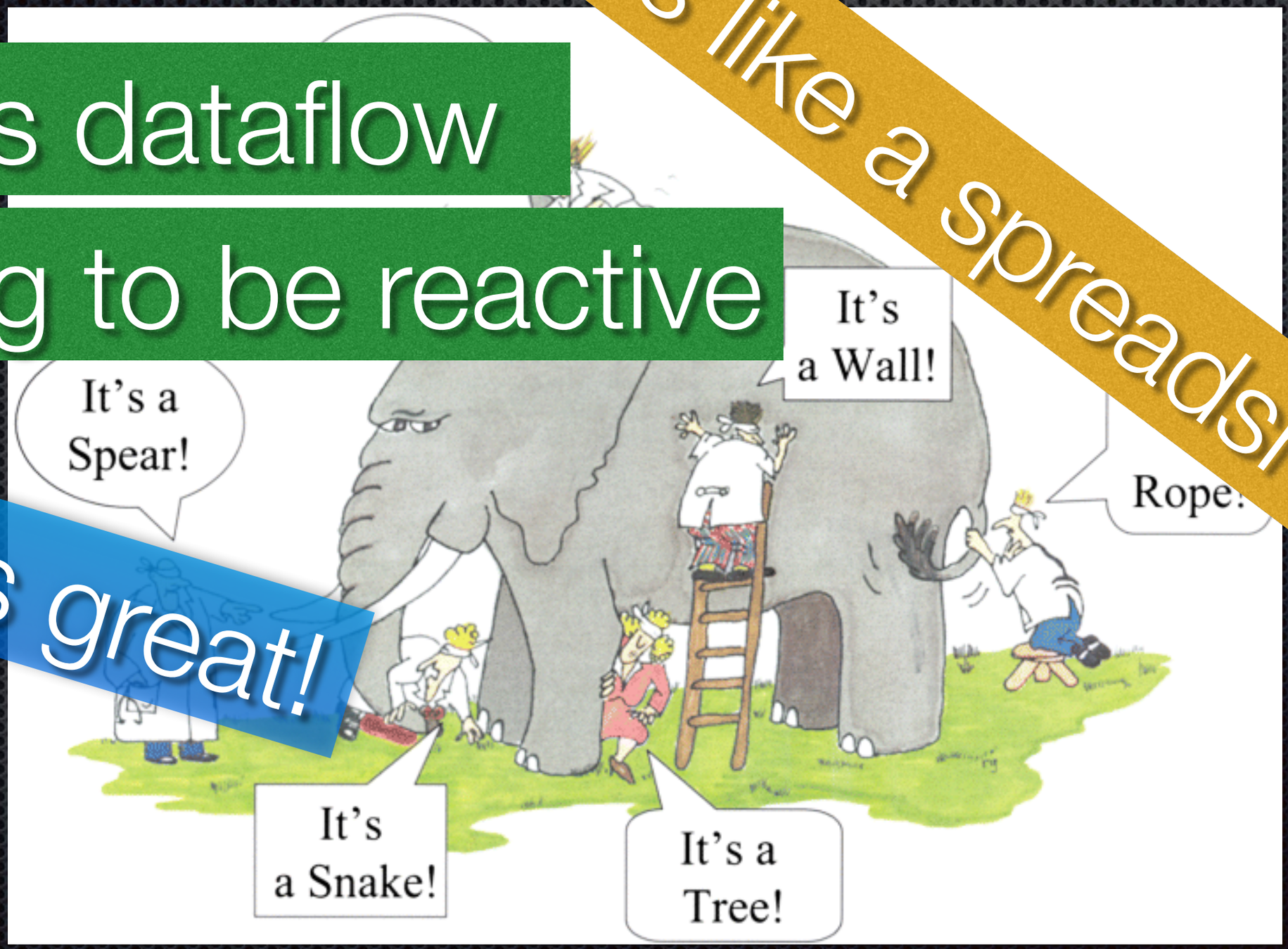
What is Reactive Programmin

It's like a spreadsheet

It's dataflow

trying to be reactive

It's great!



What is reactive?

- it's a style of system
- not an implementation style
  can be implemented in any style (OO is typical)
- it's the opposite of "functional" ("transformational")
- sometimes implemented with special kinds of dataflow (synchronous dataflow)

What is “reactive”?

- an implementation style
- dataflow
- but not the synchronous dataflow used to implement reactive systems

What is “reactive”?

- dataflow

Dataflow

- ls | wc

Dataflow

- Bentley's Challenge: most frequent words

Don Knuth

Doug McIlroy


```
tr -cs A-Za-z '\n' |
  tr A-Z a-z |
  sort |
  uniq -c |
  sort -rn |
  sed ${1}q
```

Dataflow

- ls | wc

OO Dataflow?

- Objects instead of bytes
- Messages instead of pipes
  -(void)writeObject:anObject;
- Object/method filters instead of processes

OO Dataflow!

```
38
39 @protocol Streaming
40
41 -(void)writeObject:anObject;
42
43 @end
44
45
46 @interface MPWStream : MPWObject<Streaming>
47
48 @property (nonatomic, strong) id <Streaming> target;
49
50
```

Example: Obj-Streams

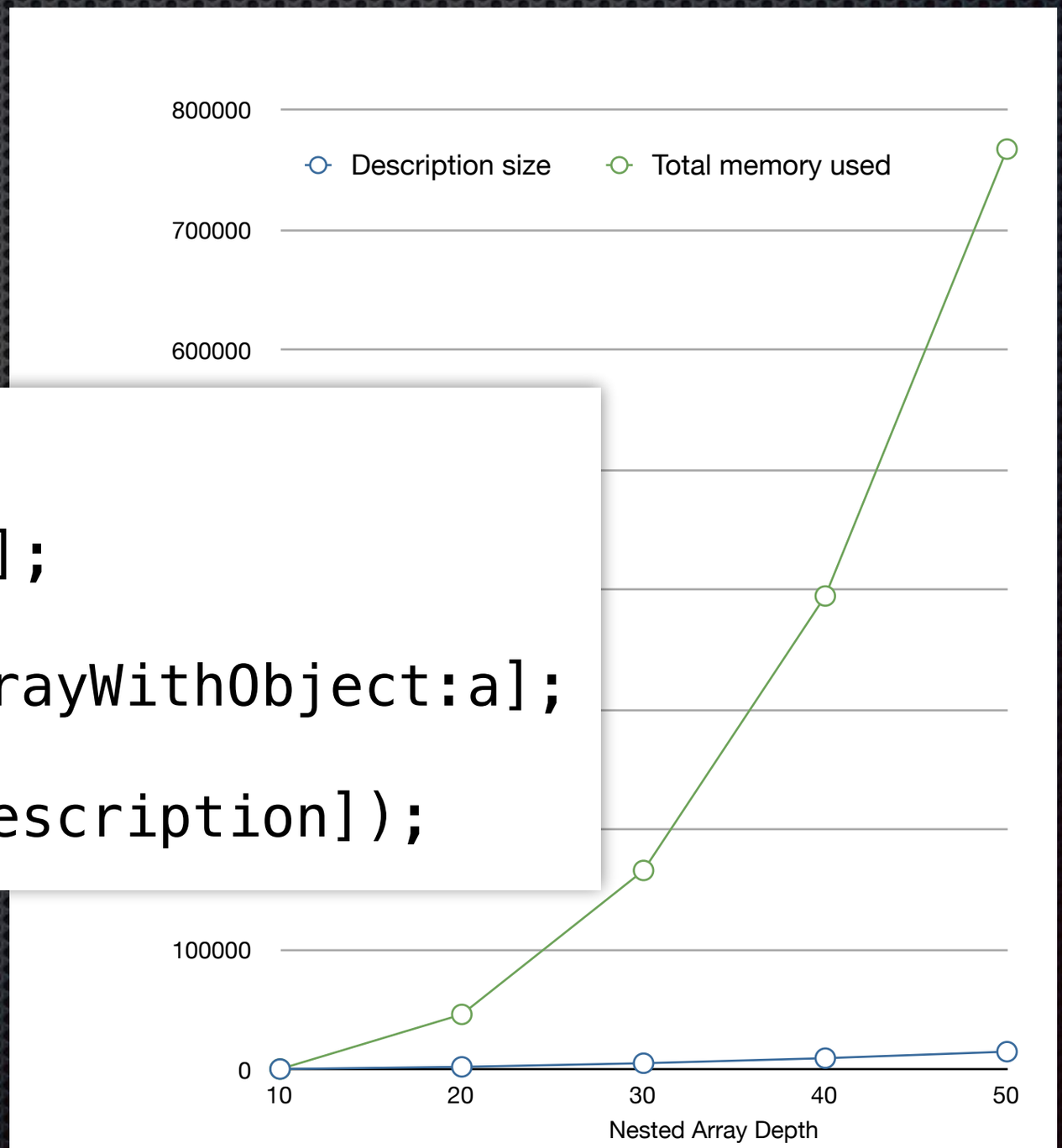
- -[NSArray description]

```
NSMutableArray *a;
a=[NSMutableArray array];
for (i=0; i<N; i++) {
   a=[NSMutableArray arrayWithObject:a];
}
NSLog(@“array: %@“,[a description]);
```

Example: Obj-Streams

- -[NSArray description]

```
NSMutableArray *a;
a=[NSMutableArray array];
for (i=0; i<N; i++) {
   a=[NSMutableArray arrayWithObject:a];
}
NSLog(@“array: %@“,[a description]);
```



Example: Obj-Streams

```
-description {
 NSMutableString *s=[NSMutableString string];
 [s appendFormat:@“<%@:%p: “,[self class],self];
 for (id obj in self ) {
    [s appendFormat: @“ %@“,[obj description];
 }
 [s appendString:@“>”];
 return s;
}
```

Example: Obj-Streams

```
-(void)describeOn:(MPWDescriptionStream*)s {
  [s writeObjectHeader:self];
  [s writeArrayContents:self];
  [s printf:@“>”];
}
```

Example: Obj-Streams

```
-(void)describeOn:(MPWDescriptionStream*)s {
 [s describeObject:self contents:^{
      [s writeArrayContents:self];
 }];
}
```

Example: Obj-Streams

```
-(void)describeOn:(MPWDescriptionStream*)s {
 [s describeObject:self contents:^{
      [s writeArrayContents:self];
 }];
}

-(NSString *)description {
  id s=[MPWDescriptionStream streamWithString];
  [s writeObject:self];
  return [s target];
}
```

Obj-Streams

- Byte-streams, printf: into an NSTextView
- flatten, all manner of serialisation
- URL fetching
- http://github.com/mpw/MPWFoundation

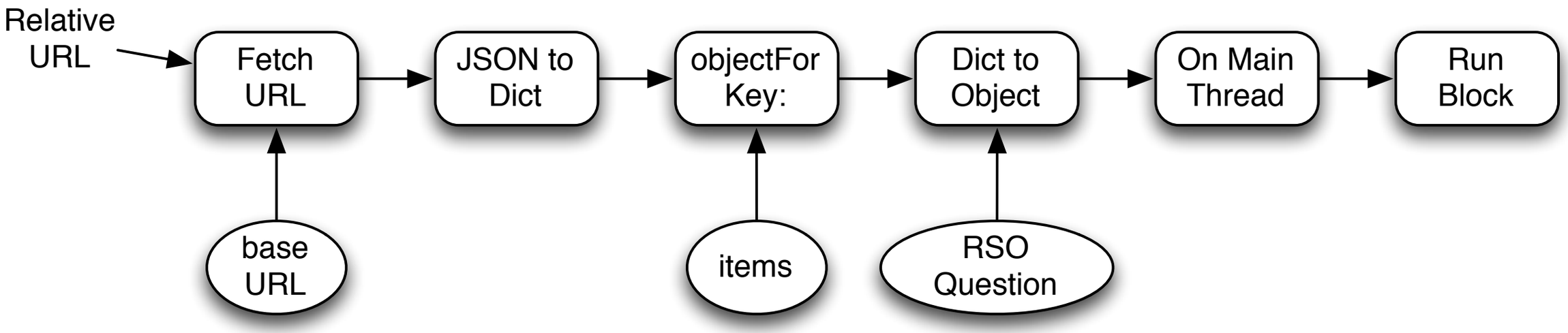
Example: RAC-Fetch

```objc
- (RACSignal *)fetchQuestionsWithTag:(NSString *)tag
{
    NSString *relativeUrl = [self createRelativeURLWithTag:tag];
    @weakify(self);
    RACSignal *signal = [RACSignal createSignal:^RACDisposable *(id<RACSubscriber> subscriber) {
        @strongify(self);
        NSURL  *fetchQuestionURL = [NSURL URLWithString:relativeUrl relativeToURL:self.baseUrl];
        NSURLSessionDataTask *task = [self.client dataTaskWithURL:fetchQuestionURL completionHandler:^(NSData *data,
            NSURLResponse *response, NSError *error) {
            if(error)
            {
                [subscriber sendError:error];
            }
            else if(!data)
            {
                NSDictionary *userInfo = @{NSLocalizedDescriptionKey: @"No data was received from the server."};
                NSError *dataError = [NSError errorWithDomain:RSOErrorDomain code:RSOErrorCode userInfo:userInfo];
                [subscriber sendError: dataError];
            }
            else
            {
                NSError *jsonError;
                NSDictionary *dict = [NSJSONSerialization JSONObjectWithData:data options:NSJSONReadingMutableContainers
                    error:&jsonError];

                if(jsonError)
                {
                    [subscriber sendError:jsonError];
                }
                else
                {
                    [subscriber sendNext:dict[@"items"]];
                    [subscriber sendCompleted];
                }
            }
        }];

        [task resume];

        return [RACDisposable disposableWithBlock:^{
            [task cancel];
        }];
    }];

    return signal;
}
```

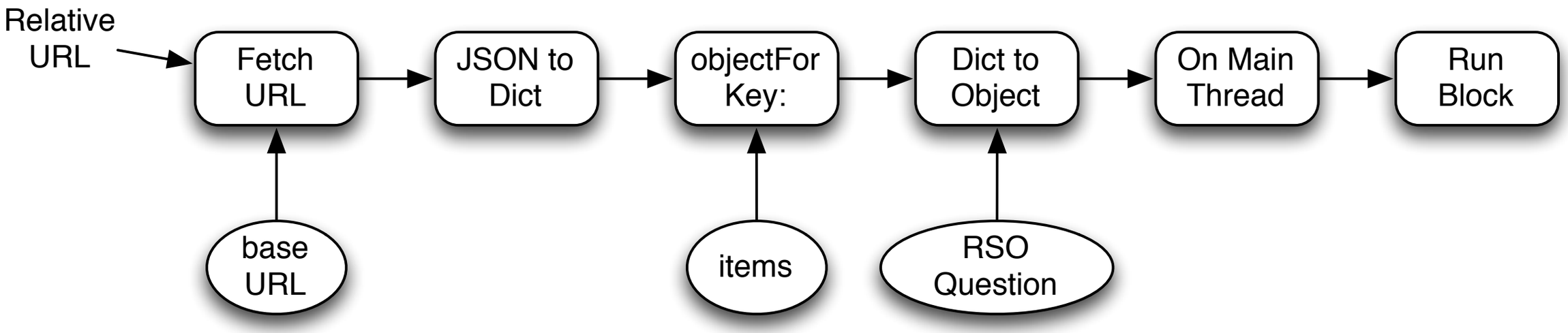
Example: Stream-Fetch

```
56 -(MPWStream*)streamSendingTo_worker:(TargetBlock)target
57 {
58     MPWURLFetchStream *fetcher;
59
60     fetcher=[[[[[MPWURLFetchStream streamWithBaseURL:[self baseUrl]
61           target:nil]
62                parseJSONWithKey:@"items" ]
63                dict2objWithClass:[RSOQuestion class]]
64                onMainThreadStream]
65                onBlock:target];
66
67     return fetcher;
68 }
```

Dataflow


Relative URL
Fetch URL
JSON to Dict
objectFor Key:
Dict to Object
On Main Thread
Run Block
base URL
items
RSO Question

Dataflow




```
URLFetchStream streamWithBaseURL: baseUrl |
                                parseJSON |
                    objectForKey:'items' |
     dict2objWithClass:(RSOQuestion class) |
                      onMainThreadStream |
                               onBlock:target.
```

Dataflow

Lucid, the Dataflow Programming Langugage[1]

William W. Wadge Edward A. Ashcroft

Naturally, we make no claims to having discovered dataflow and the dataflow approach to computation. For this, many people share the credit, people such as M. McIlroy, J. Dennis and G. Kahn. We owe a debt to the developers of UNIX,[1] who have provided practical proof that dataflow is a powerful programming technique.

```
NETWORK DIVISION.
  CONNECT PROGRAM-INPUT TO SQUARE-INPUT
  CONNECT SQUARE-OUTPUT TO AVG-INPUT
  CONNECT AVG-OUTPUT TO SQROOT-INPUT
  CONNECT SQROOT-OUTPUT TO PROGRAM-OUTPUT
```

The Lucid programmer, however, does not directly manipulate lines and filters. Instead, the Lucid user specifies the *data* to be processed and the *transformations* to be applied. Consider,

Our starting point was the observation that the information contained in the assignment statements

$$I := 1;$$
$$\vdots$$
$$I := I + 1;$$

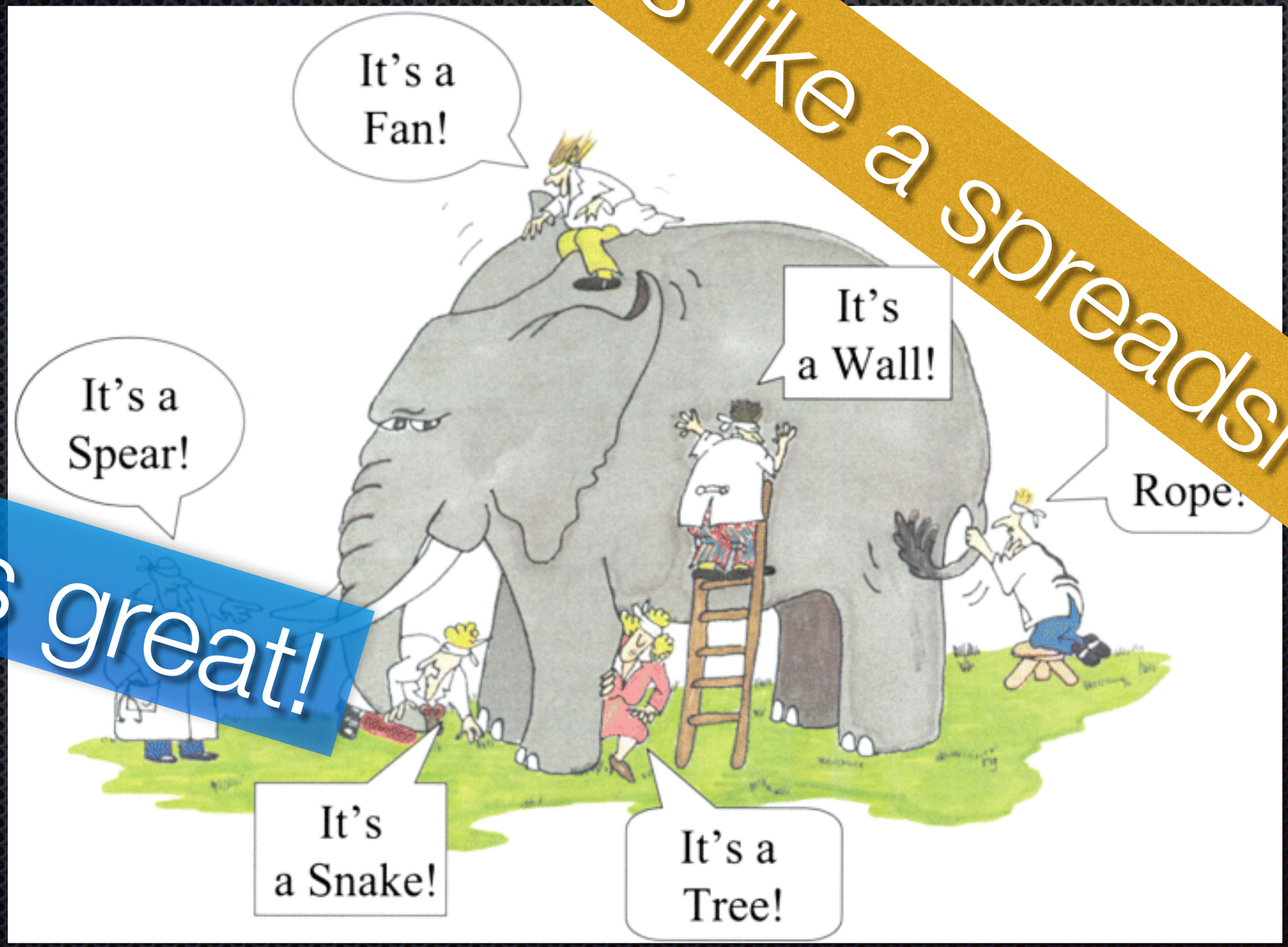
could be conveyed by the declarations

$$\texttt{first}\ I = 1;$$
$$\texttt{next}\ I = I + 1;$$

The form of these declarations suggested that `first` and `next` could be used as operations and that these operations could be applied to expressions as well as simple identifiers. This in turn opened up the possibility that intuitively true equations like $\texttt{next}(x + y) = \texttt{next}(x) + \texttt{next}(y)$ could be used to reason algebraically about dynamic aspects of programs.

What is Reactive Programmin

It's like a spreadsheet



It's great!

Spreadsheet

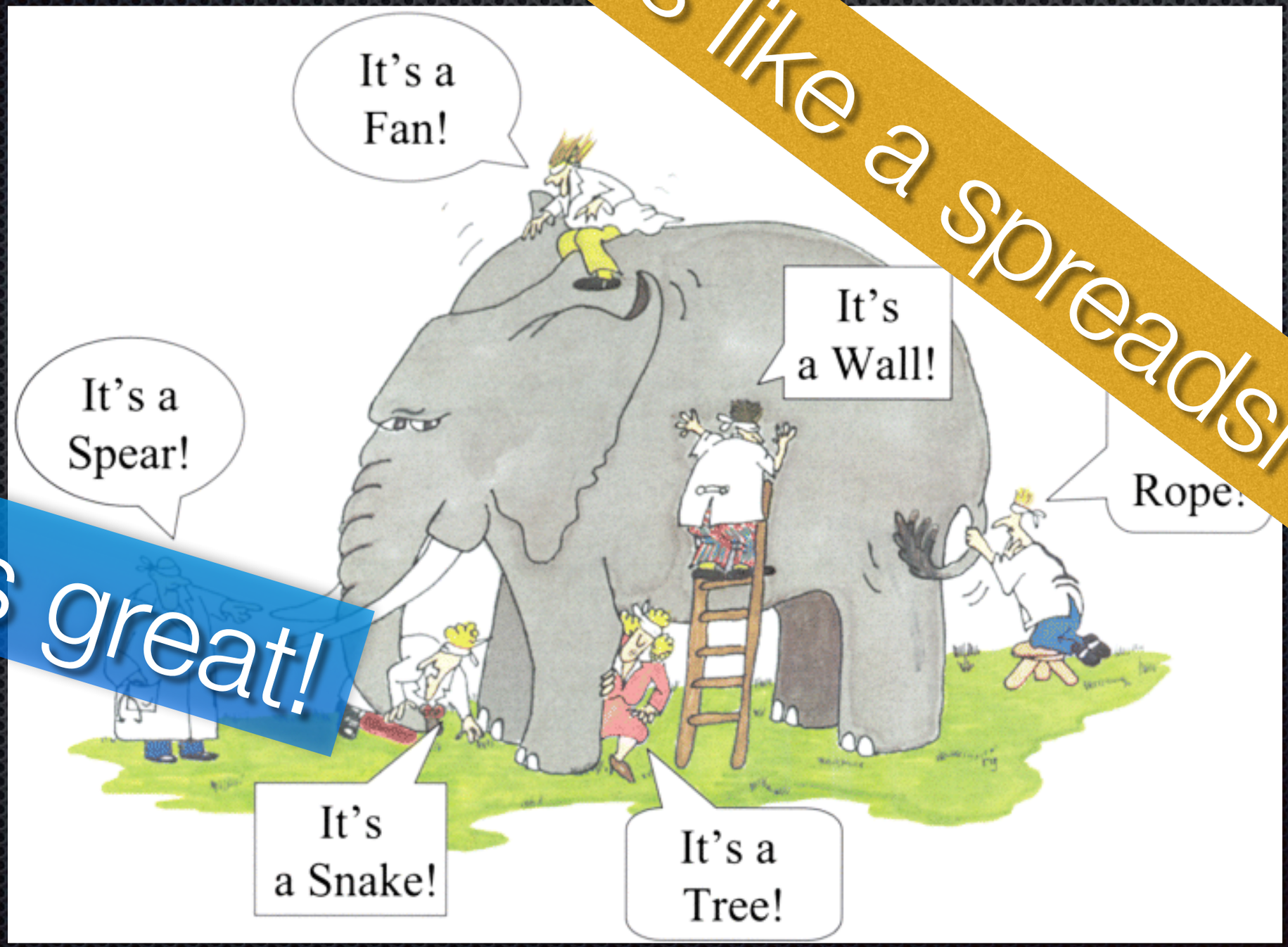
```
RAC(self, createEnabled) = [RACSignal
    combineLatest:@[ RACObserve(self, password),
                     RACObserve(self, passwordConfirmation) ]
    reduce:^(NSString *password, NSString *passwordConfirm) {
        return @([passwordConfirm isEqualToString:password]);
    }];
```

Table 1

		Enabled
Password	dasdasdasd	B2 = B3
Repeat	dasdasda	

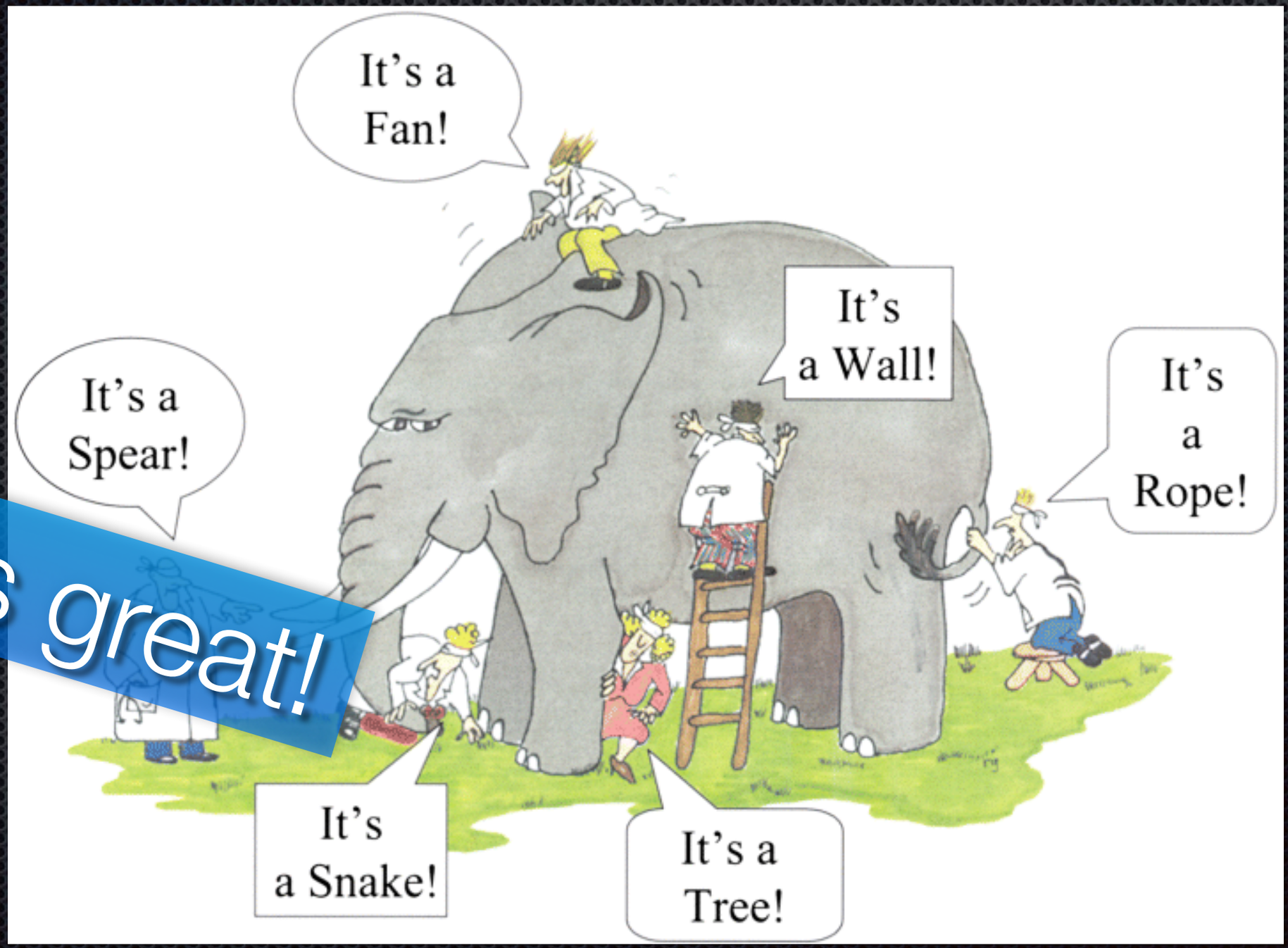
What is Reactive Programmin

It's like a spreadsheet



It's great!

What is Reactive Programming?



It's great!

Dataflow Constraints

- aka “Spreadsheet Constraints”
- Typically involve a solver (e.g. DeltaBlue)
- When a variable changes, solver creates a “plan”
- This plan corresponds to a reactive program

Dataflow Constraints

Generating Reactive Programs for Graphical User Interfaces from Multi-way Dataflow Constraint Systems

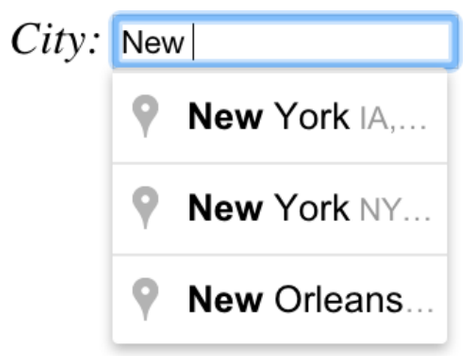
Gabriel Foust
Texas A&M University, TX, USA
gfoust@cse.tamu.edu

Jaakko Järvi
Texas A&M University, TX, USA
jarvi@cse.tamu.edu

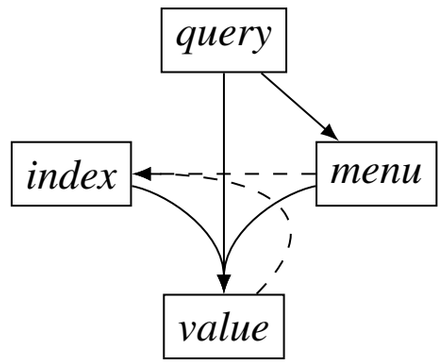
Sean Parent
Adobe Systems, Inc.
sparent@adobe.com

Abstract

For a GUI to remain responsive, it must be able to schedule lengthy tasks to be executed asynchronously. In the traditional approach to GUI implementation—writing functions to handle individual user events—asynchronous programming easily leads to defects. Ensuring that all data dependencies are respected is difficult when new events arrive while prior events are still being handled. Reactive programming techniques, gaining popularity in GUI programming, help since they make data dependencies explicit and enforce them automatically as variables' values change. However,



(a) An auto-complete text box.



(b) The data dependencies.

Figure 1: An example of an auto-complete text box, and a diagram

Dataflow Constraints I

```
loginButton.enabled =
    [passwordField.stringValue isEqual:
       repeatField.stringValue];
```

Dataflow Constraints I

```
loginButton.enabled =
    [passwordField.stringValue isEqual:
       repeatField.stringValue];
```

```
loginButton/enabled :=
     passwordField/stringValue =
       repeatField/stringValue.
```

Dataflow Constraints I

```
loginButton/enabled :=
     passwordField/stringValue =
       repeatField/stringValue.
```

Dataflow Constraints I

```
loginButton/enabled |=
     passwordField/stringValue =
       repeatField/stringValue.
```

Dataflow Constraints I

```
RAC(self, createEnabled) = [RACSignal
    combineLatest:@[ RACObserve(self, password),
                     RACObserve(self, passwordConfirmation) ]
    reduce:^(NSString *password, NSString *passwordConfirm) {
        return @([passwordConfirm isEqualToString:password]);
    }];
```

```
loginButton/enabled |=
     passwordField/stringValue =
       repeatField/stringValue.
```

Temperature Converter Model Constraints

```
celsius    |= fahrenheit * 5/9 - 32.
fahrenheit |= (celsius + 32) * 9/5.
```

Temperature Converter
Add UI

```
celsiusTextField/intValue     =|= celsius.
fahrenheitTextField/intValue =|= fahrenheit.


celsius     |= fahrenheit * 5/9 - 32.
fahrenheit |= (celsius + 32) * 9/5.
```

Temperature Converter Add Persistence

```
celsius                 := defaults:celsius.
defaults:celsius        |= celsius.

celsiusTextField/intValue    =|= celsius.
fahrenheitTextField/intValue =|= fahrenheit.


celsius    |= fahrenheit * 5/9 - 32.
fahrenheit |= (celsius + 32) * 9/5.
```

Temperature Converter Add Kelvin

```
celsius                 := defaults:celsius.
defaults:celsius        |= celsius.

celsiusTextField/intValue     =|= celsius.
fahrenheitTextField/intValue  =|= fahrenheit.
kelvinTextField/intValue      =|= kelvin.

celsius     |= fahrenheit * 5/9 - 32.
fahrenheit  |= (celsius + 32) * 9/5.
kelvin      |= celsius - 273.
celsius     |= kelvin + 273.
```

Temperature Converter Group

```
memory-model          := persistence.
persistence           |= memory-model.

celsiusTextField/intValue    =|= celsius.
fahrenheitTextField/intValue =|= fahrenheit.
kelvinTextField/intValue     =|= kelvin.

celsius     |= fahrenheit * 5/9 - 32.
fahrenheit  |= (celsius + 32) * 9/5.
kelvin      |= celsius - 273.
celsius     |= kelvin + 273.
```

Temperature Converter Group

```
memory-model        := persistence.
persistence         |= memory-model.

ui                 =|= memory-model.


celsius     |= fahrenheit * 5/9 - 32.
fahrenheit  |= (celsius + 32) * 9/5.
kelvin      |= celsius - 273.
celsius     |= kelvin + 273.
```

Temperature Converter Group

```
memory-model          := persistence.
persistence           |= memory-model.

ui                   =|= memory-model.


celsius              =|= fahrenheit * 5/9 - 32.
celsius              =|= kelvin + 273.
```

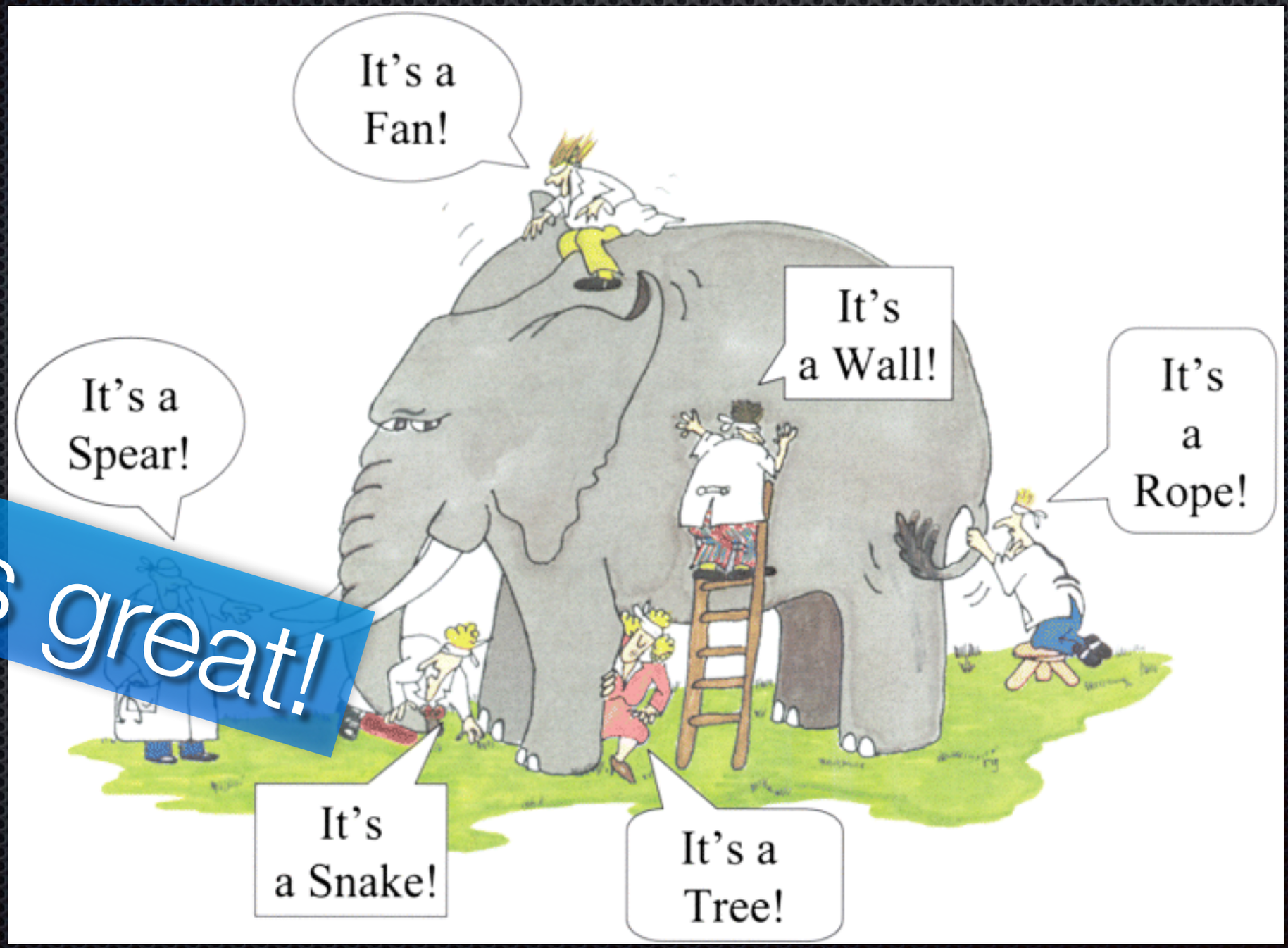
Temperature Converter Group

```
memory-model          := persistence.
persistence           |= memory-model.

ui                   =|= memory-model.

memory-model         =|= “consistency”
```

Wunderlist Architecture

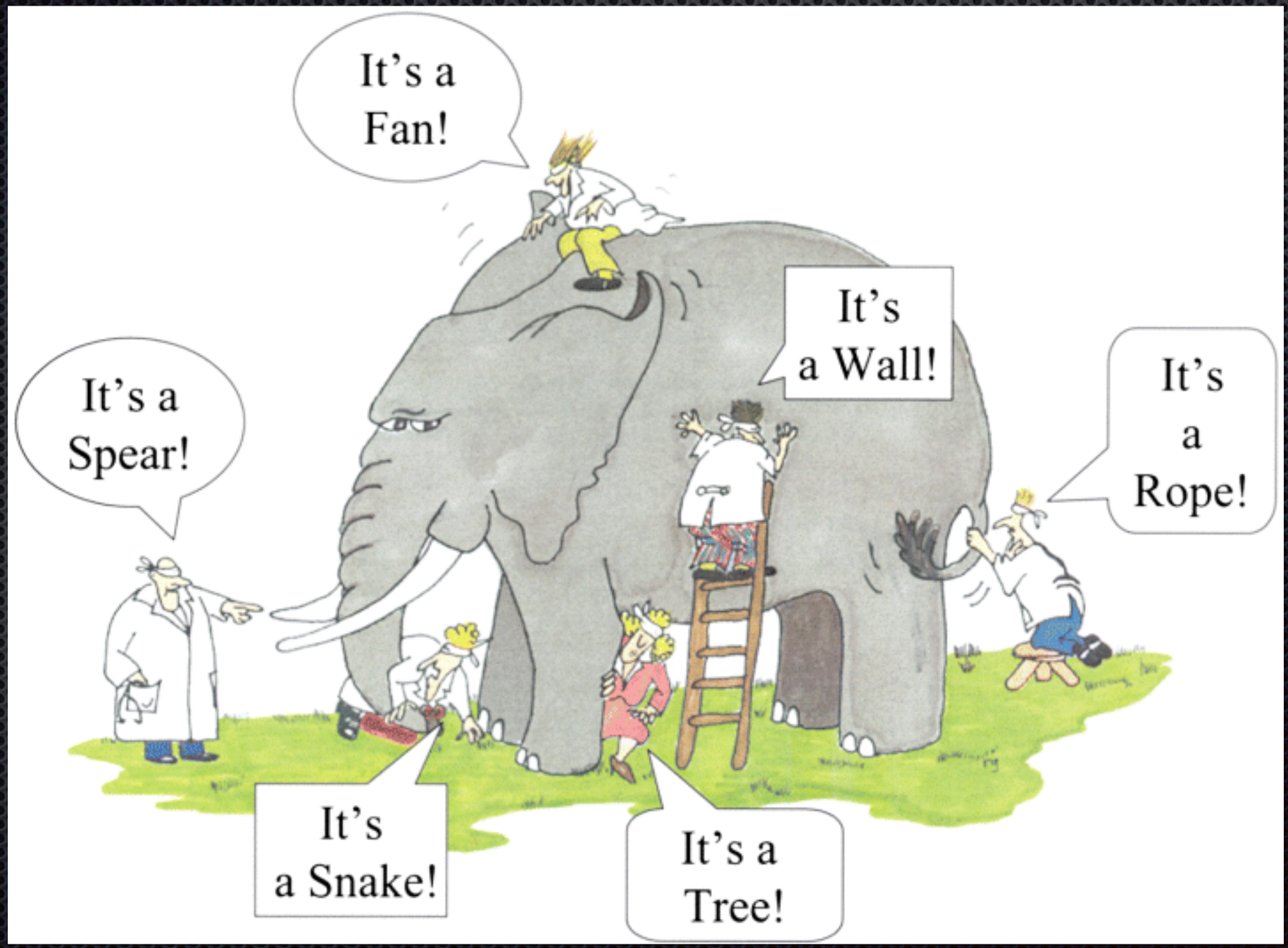
```
memory-model          := persistence.
persistence           |= memory-model.

ui                   =|= memory-model.

memory-model         =|= backend
```

What is Reactive Programming?



It's great!

What is Reactive Programming?



Reactive Programming Dataflow Constraints

Marcel Weiher @mpweiher

- Dataflow! (for example with MPWStream)
- Dataflow constraints compile to reactive programs