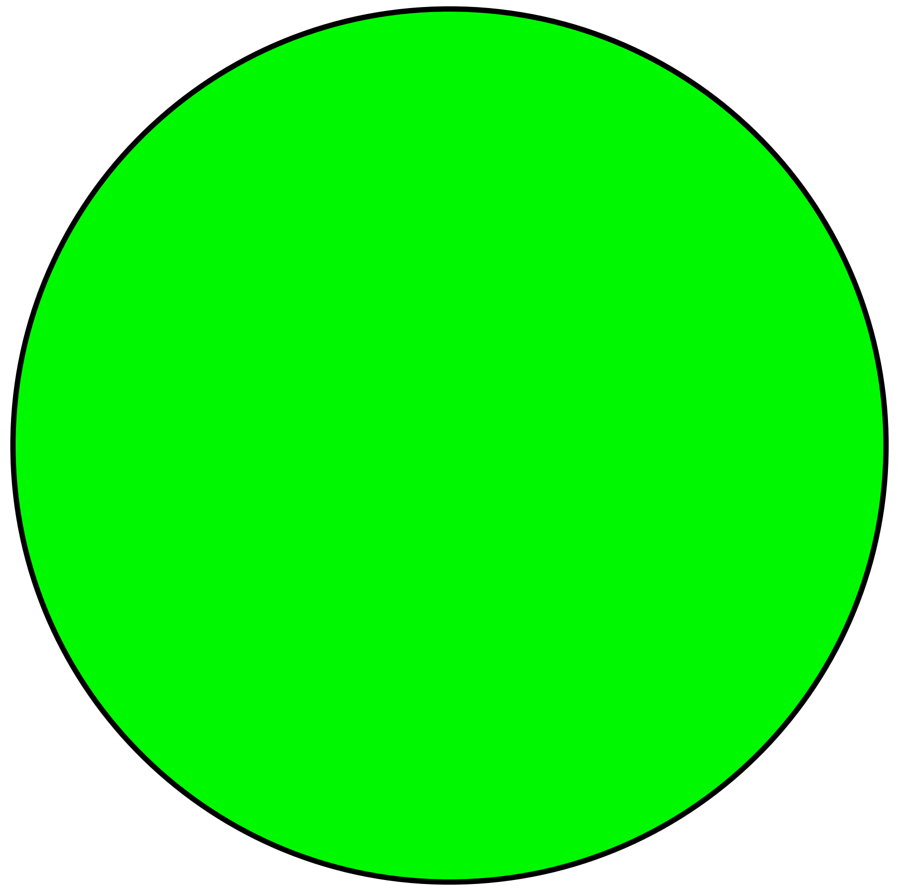
CMPU 101 § 54 · Computer Science I

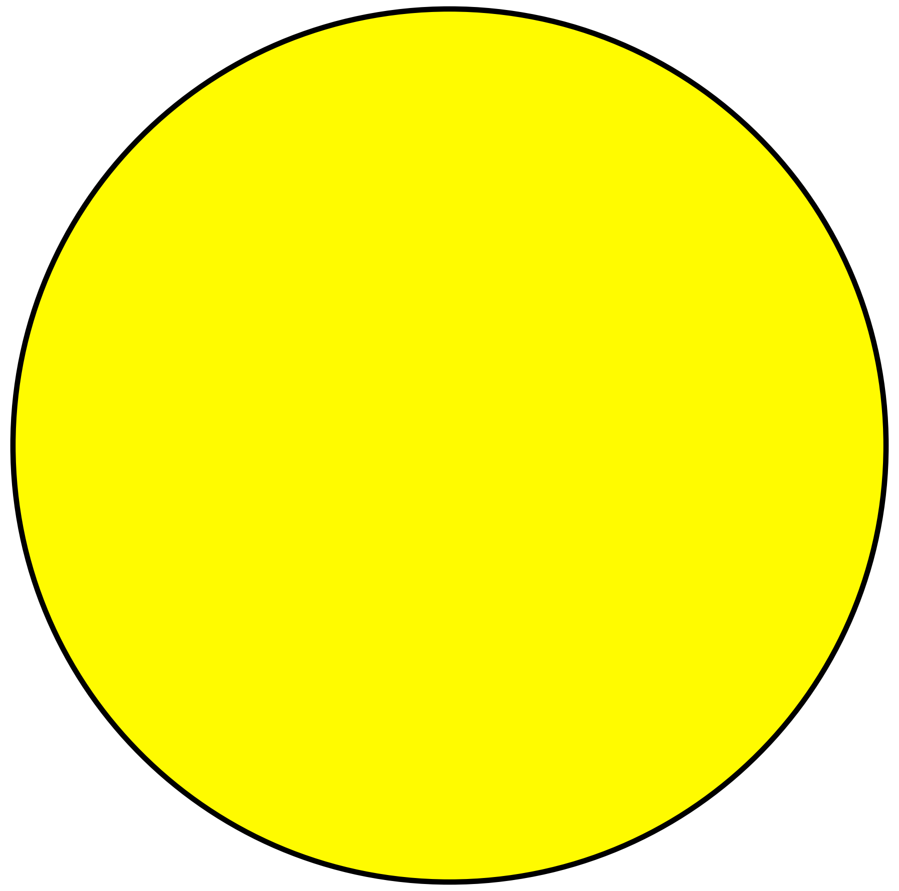
Reactive Programs

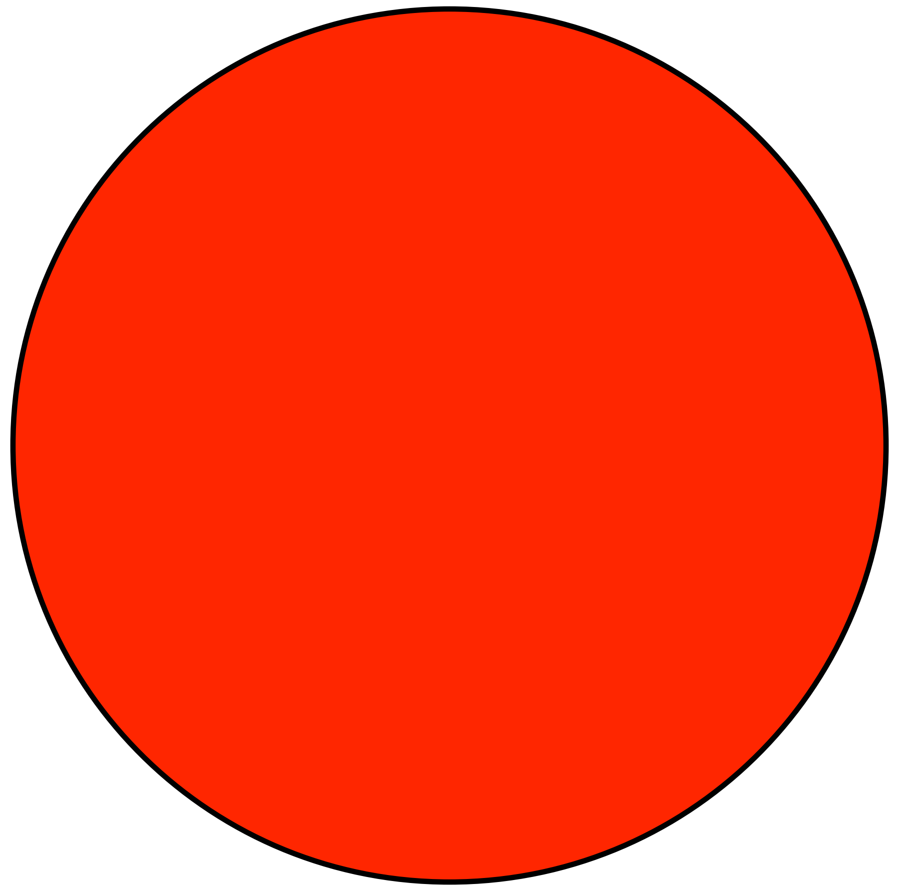
1 March 2023

Where are we?

Where are we?
Traffic-light world







All traffic lights are the same size and position on the screen.

What distinguishes them?

Asking this helps us think about data

All traffic lights are the same size and position on the screen.

How do we get from one to the other?

Asking this helps us think about functions.
We need more(?), but this is a good start.
i.e. not just how, but when.

Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

Data

Data definition

Examples

Template

Functions

Signature

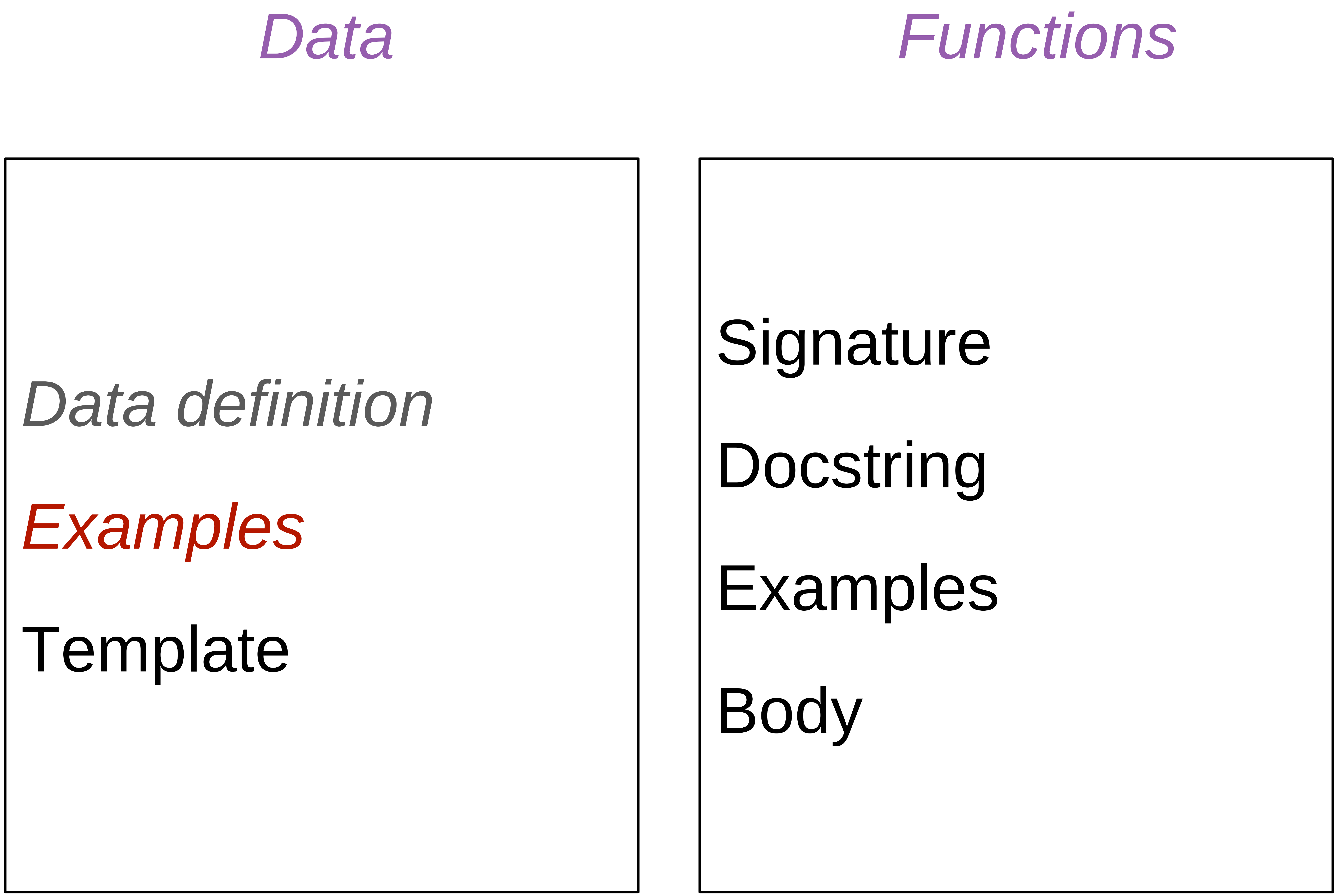
Docstring

Examples

Body

```
data TrafficLight:
  ...
end
```

```
data TrafficLight:
  | green
  | yellow
  | red
end
```



Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

```
data TrafficLight:
  | green
  | yellow
  | red
end

TL-GREEN = green
TL-YELLOW = yellow
TL-RED = red
```

For this data definition, the examples are so trivial we can skip them, but you saw in the pipeline lab how helpful it can be to have examples when you have a lot of possibilities!

```
data TrafficLight:
  | green
  | yellow
  | red
end
```

Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

```
data TrafficLight:
  | green
  | yellow
  | red
end
```

```
data TrafficLight:
  | green
  | yellow
  | red
end
#|
fun trafficlight-fun(tl :: TrafficLight) -> ...:
|#
```

```
data TrafficLight:
  | green
  | yellow
  | red
end
#|
fun trafficlight-fun(tl :: TrafficLight) -> ...:
  doc: "TrafficLight template"
  cases (TrafficLight) tl:
    | green => ...
    | yellow => ...
    | red => ...
  end
where:
  trafficlight-fun(green) is ...
  trafficlight-fun(yellow) is ...
  trafficlight-fun(red) is ...
end |#
```

Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

As we saw* last class, Pyret has a mechanism for supporting interactive programs, called a **reactor.** It allows us to model the real word**

To use it, first write

```
include reactors
```

*Well, we didn't *actually* see it, but just go with it ☺

**Not the television show!

```
reactor:
  init: initial-state
  to-draw: draw-function
  event-type: event-function
end
```

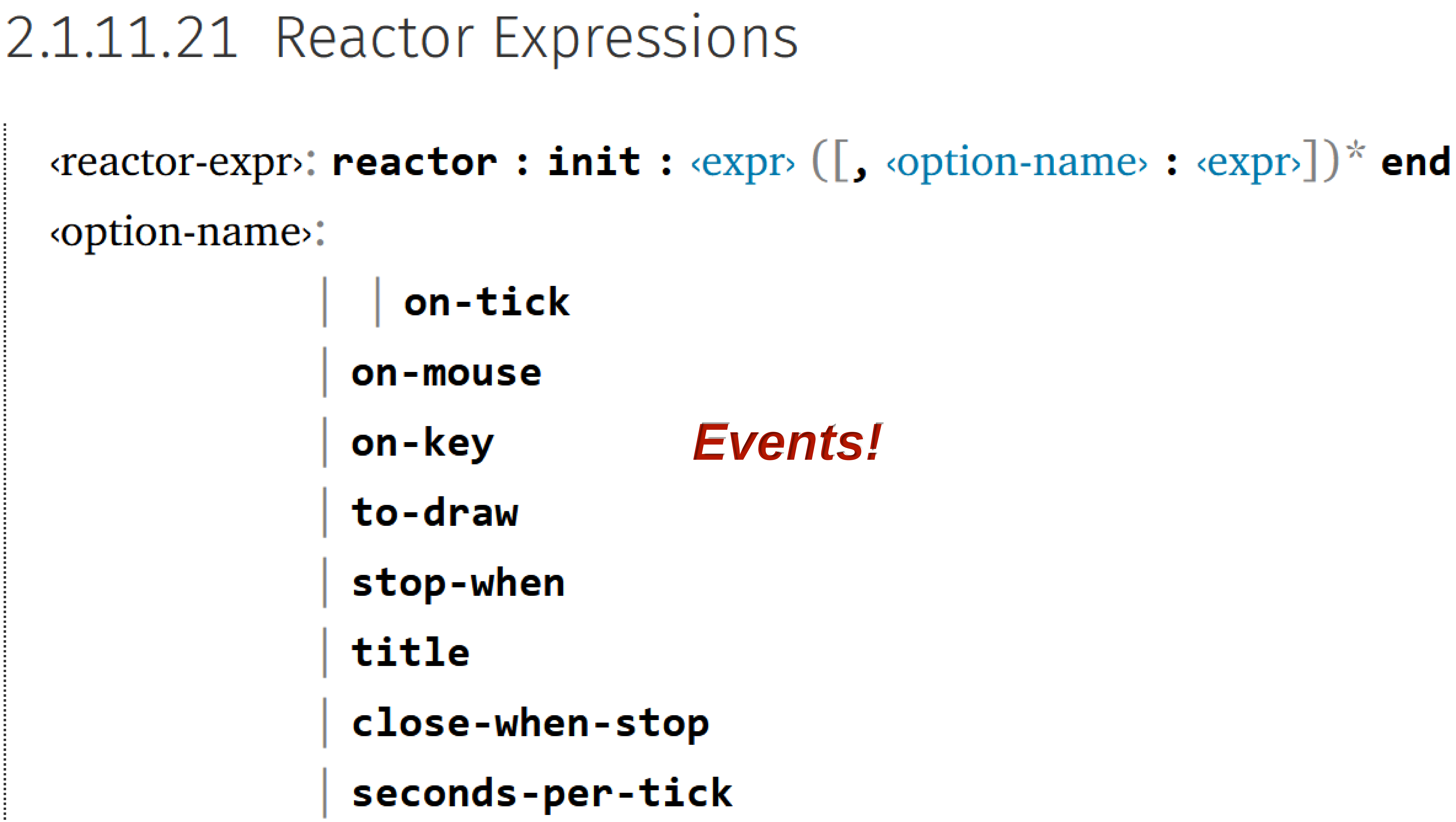
Event-type allows us to ask **“when?”** do we want to call the **draw-function**. The 4’th dimension exists!

“reactor?” this allows us our program to react to events. What kind of events? Something like…

```
reactor:
  init: initial-state
  to-draw: draw-function
  event-type: event-function
end
```

Less nuclear reactor; more person-that-reacts to something.

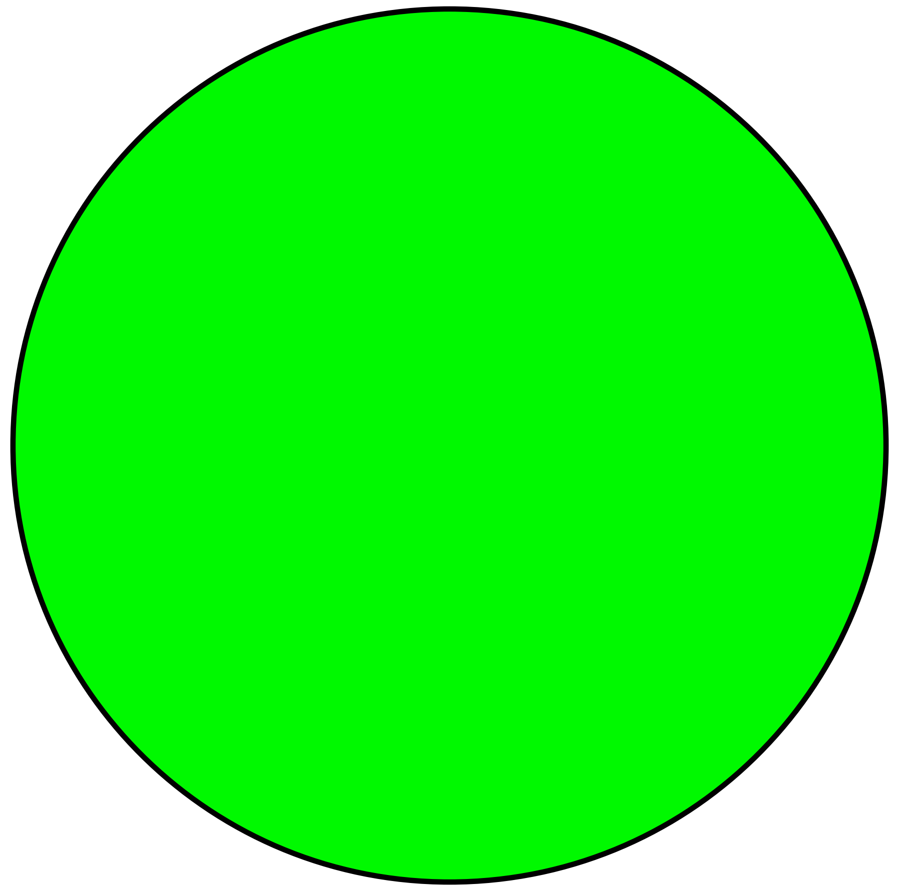
reactor puts all the pieces together to start things up and allows the program to react to events...



2.1.11.21 Reactor Expressions

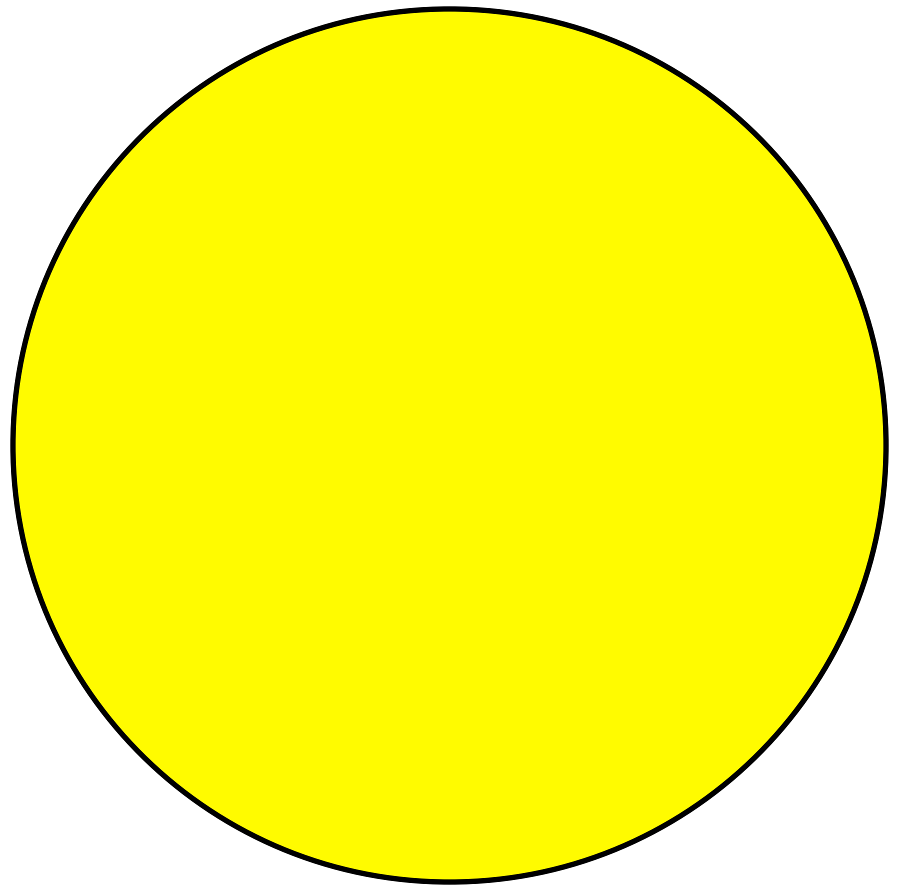
```
‹reactor-expr›: reactor : init : ‹expr› ([, ‹option-name› : ‹expr›])* end
‹option-name›:
              |  | on-tick
              | on-mouse
              | on-key
              | to-draw
              | stop-when
              | title
              | close-when-stop
              | seconds-per-tick
```

Events!

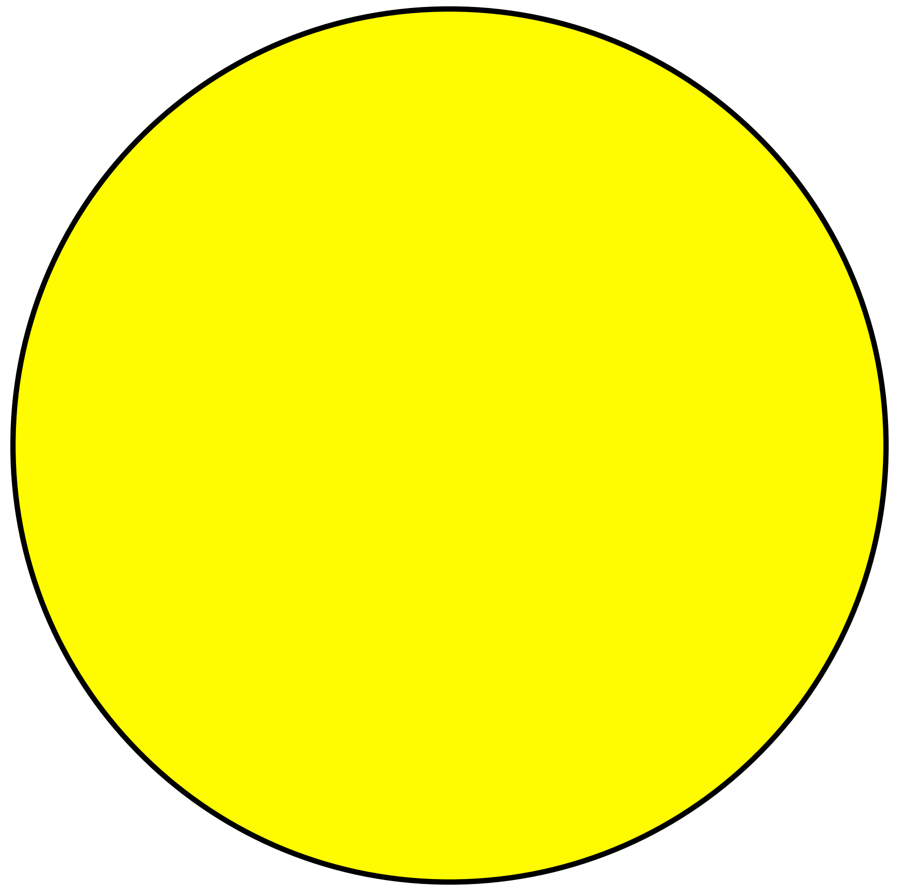


initial state

some event happens…



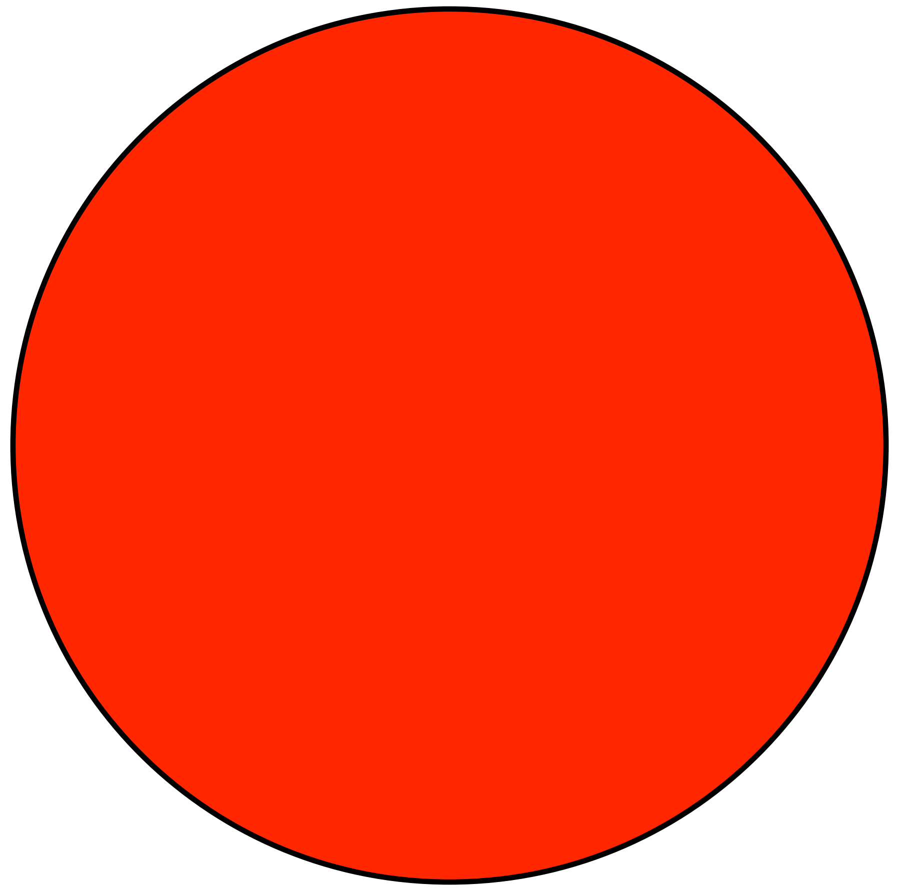
next state



~~next state~~

now the current state

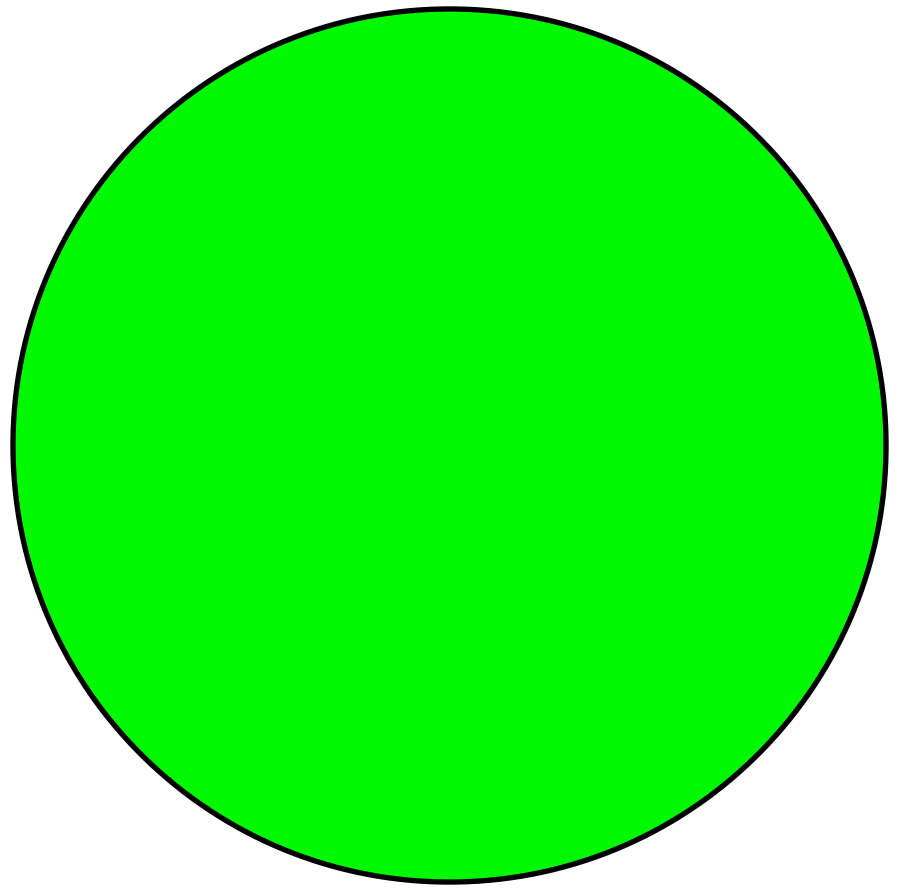
some event happens…



~~next state~~

now the current state

some event happens…



~~*next state*~~

now the current state

```
reactor:
  init: initial-state,
  to-draw: draw-function,
  event-type: event-function
end
```

```
reactor:
  init: green,
  to-draw: draw-function,
  event-type: event-function
end
```

```
reactor:
  init: green,
  to-draw: draw-light,
  event-type: event-function
end
```

We haven’t written this; add it to our wishlist!

```
reactor:
  init: green,
  to-draw: draw-light,
  on-tick: next-light
end
```

Another function for the wishlist!

So far…

```
# TrafficLight data
# - definition
# - examples
# - template

# define reactor

# Wishlist:
# - fun draw-light...
# - fun next-light...
```

Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

```
fun draw-light(tl :: TrafficLight) -> Image:
  ...
end
```

```
fun draw-light(tl :: TrafficLight) -> Image:
  ...
end
```

```
fun next-light(tl :: TrafficLight) -> TrafficLight:
  ...
end
```

Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

```
fun draw-light(tl :: TrafficLight) -> Image:
  doc: "Draw a circle of the given color, rendering a traffic light"
  ...
end
```

```
fun next-light(tl :: TrafficLight) -> TrafficLight:
  ...
end
```

```
fun draw-light(tl :: TrafficLight) -> Image:
  doc: "Draw a circle of the given color, rendering a traffic light"
  ...
end
```

```
fun next-light(tl :: TrafficLight) -> TrafficLight:
  doc: "Produce the next light in the sequence green, yellow, red"
  ...
end
```

Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

```
fun draw-light(tl :: TrafficLight) -> Image:
  doc: "Draw a circle of the given color, rendering a traffic light"
  ...
where:
  draw-light(green) is circle(20, "solid", "green")
  draw-light(yellow) is circle(20, "solid", "yellow")
  draw-light(red) is circle(20, "solid", "red")
end
```

```
fun next-light(tl :: TrafficLight) -> TrafficLight:
  doc: "Produce the next light in the sequence green, yellow, red"
  ...
end
```

```
fun draw-light(tl :: TrafficLight) -> Image:
  doc: "Draw a circle of the given color, rendering a traffic light"
  ...
where:
  draw-light(green) is circle(20, "solid", "green")
  draw-light(yellow) is circle(20, "solid", "yellow")
  draw-light(red) is circle(20, "solid", "red")
end
```

```
fun next-light(tl :: TrafficLight) -> TrafficLight:
  doc: "Produce the next light in the sequence green, yellow, red"
  ...
where:
  next-light(green) is yellow
  next-light(yellow) is red
  next-light(red) is green
end
```

Data

Data definition

Examples

Template

Functions

Signature

Docstring

Examples

Body

Starter code:

tinyurl.com/2023-03-01-tl-starter

Code:

tinyurl.com/2023-03-01-tl

Screensaver

DVD
VIDEO

Code:

tinyurl.com/2023-03-01-bounce

Acknowledgments

This lecture incorporates material from:

W. Daniel Hillis, *The Pattern on the Stone*

Marc Smith, Vassar College

Laney Strange, Northeastern University