Functional Reactive Programming
:
Делаем жизнь проще

Trdat Mkrtchyan @zeffirsky github/zefirka
Parcsis
MoscowJS Meetup 19

(Функциональное)

```
                Can't embed values when interpreti
              tion code-in-interpreter (embedded-lisp

            (c s)
          rmat s "Can't embed code when interpreting. Code:

        un emit-element-body (processor tag body)
      (when (block-element-p tag)
```

(**Плюсы**

(Иммутабельные данные)

(Отсутствие состояния)

(Лень как стратегия вычислений)

(Частничное применение и карринг)

(? Рекурсия вместо итерации))

Декларативный подход

*(**Минсуы** (*

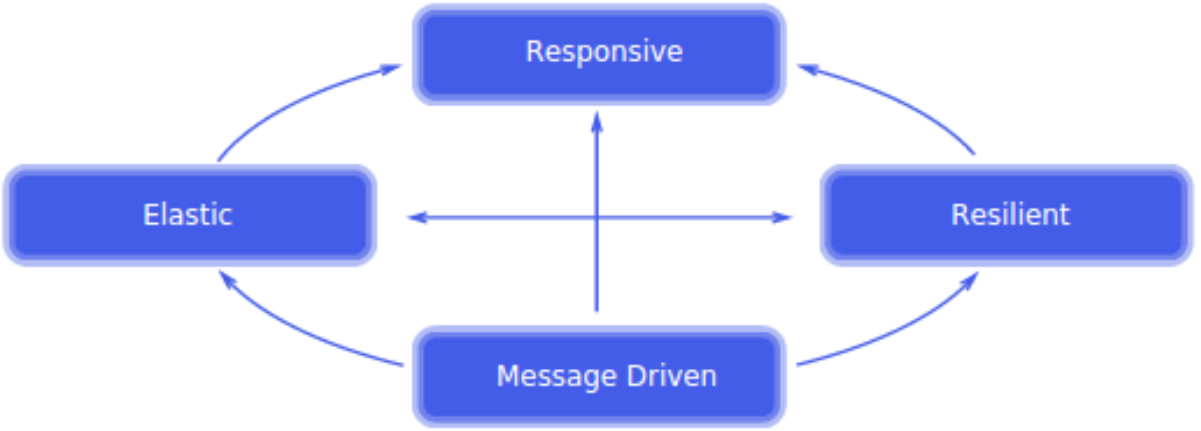
(Производительность)

(Комьюнити слишком маленькое)

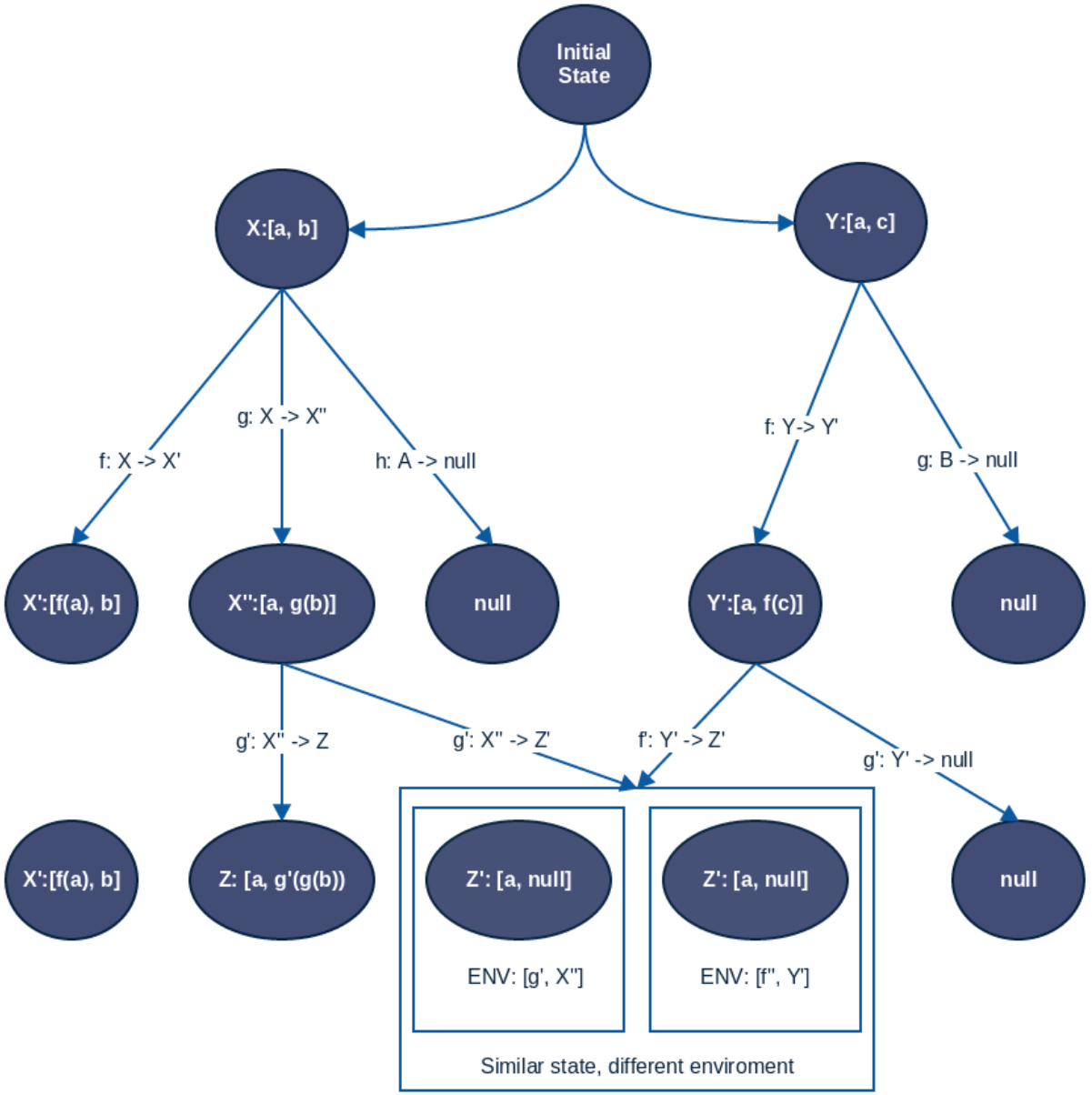
(Состояние есть, но оно скрыто))

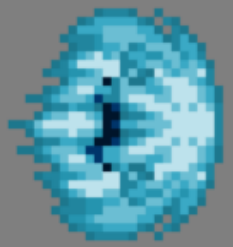
(Реактивное)

www.reactivemanifesto.org



Асинхронность + Дерево состояний = Непредсказуемость

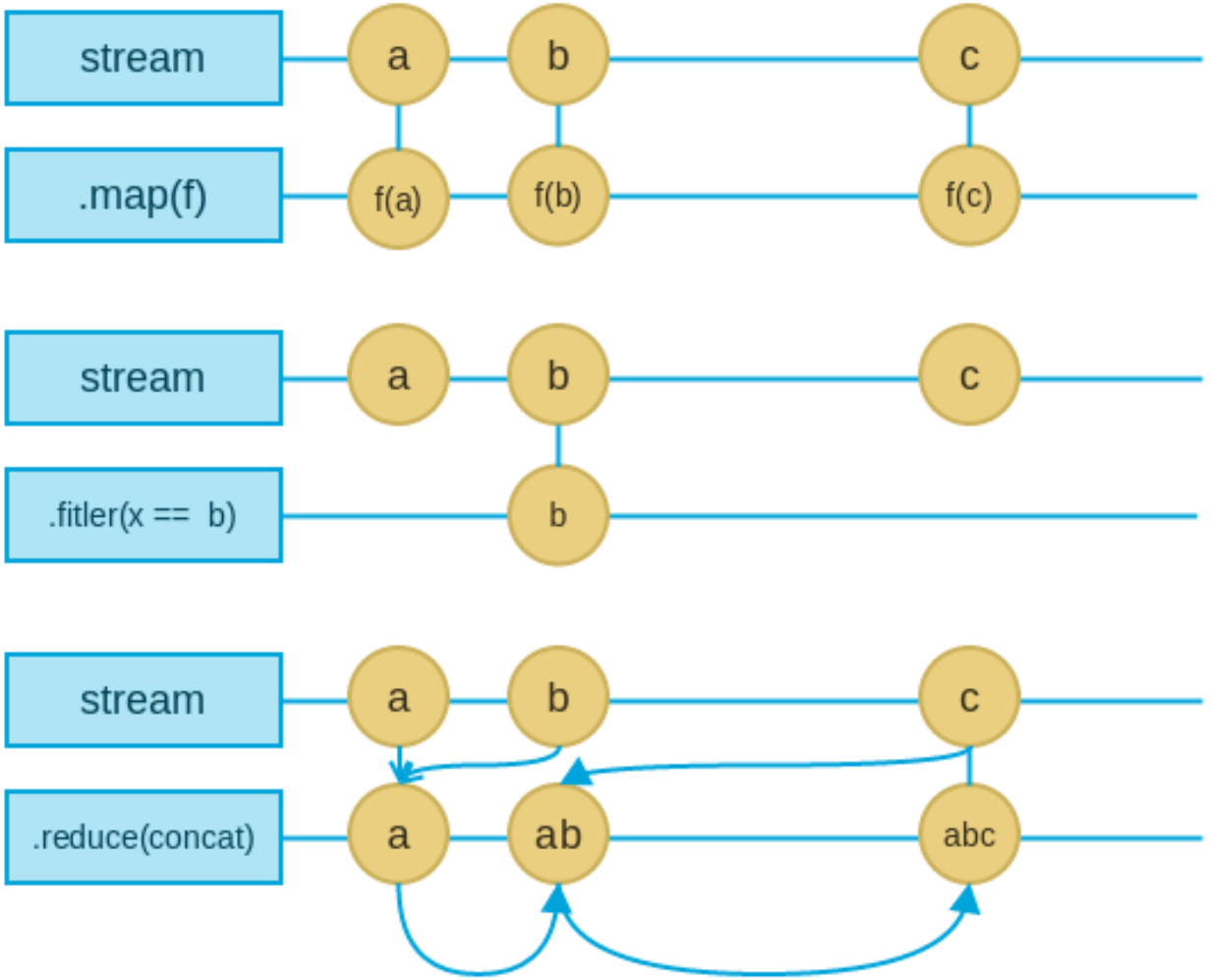




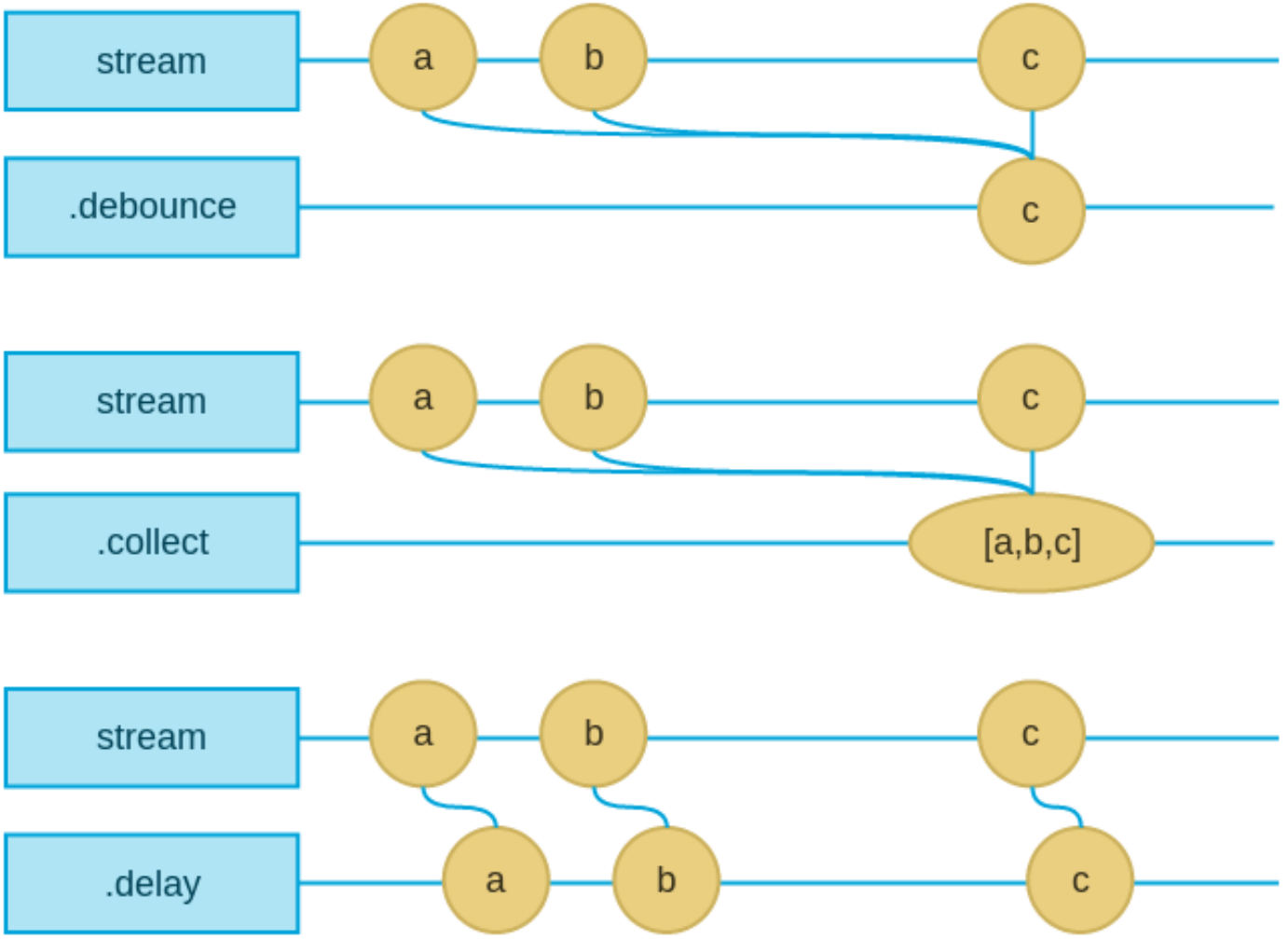
```
function doSomething(params){
  $.get(url, function(result){
    setTimeout(function(){
      startAsyncProcess(function(){
        $.post(url, function(response){
          if(response.good){
            setStateasGoodResponse(function(){
              console.log('Hooray!')
            });
          }
        });
      });
    });
  });
}
```

www.zefirka.github.io/MoscowJS

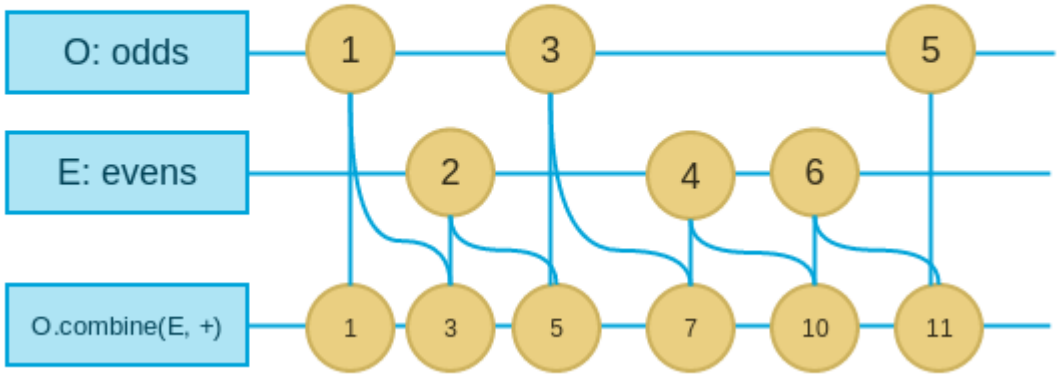
(map / filter / reduce)

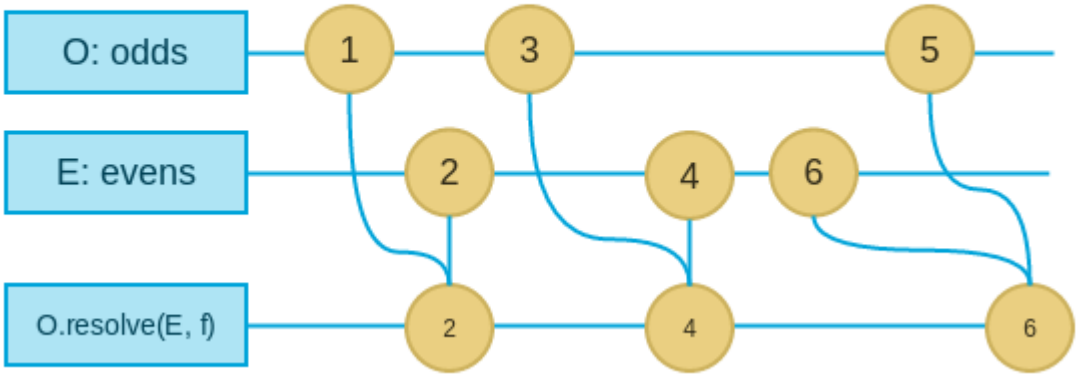


(Timing)

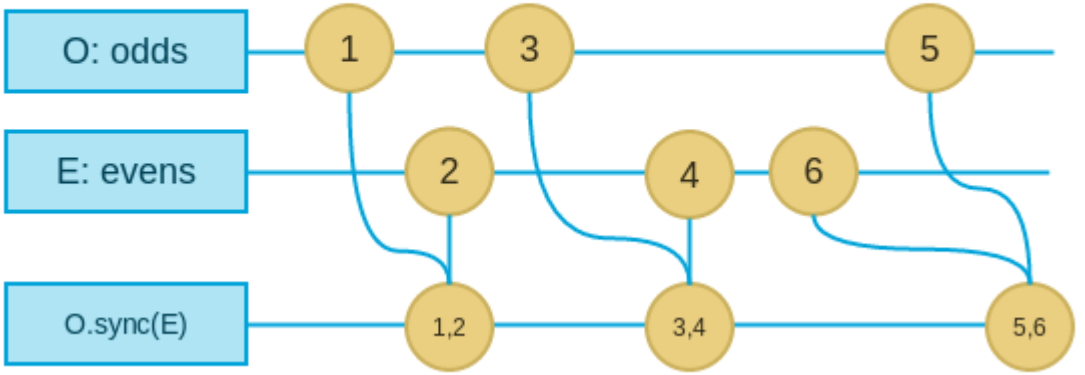

stream
a
b
c
.debounce
c
stream
a
b
c
.collect
[a,b,c]
stream
a
b
c
.delay
a
b
c

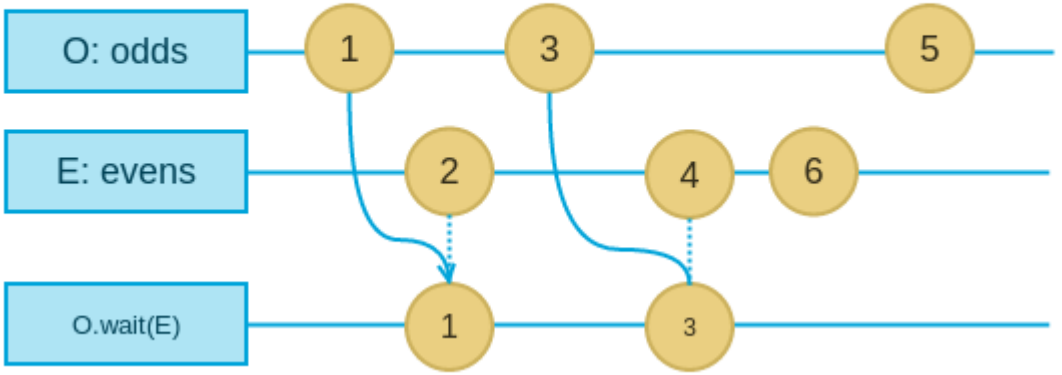
(Combining)


O: odds
E: evens
O.combine(E, +)
1
3
5
2
4
6
1
3
5
7
10
11


O: odds
E: evens
O.resolve(E, f)
1
3
5
2
4
6
2
4
6

(Composing)


O: odds
1
3
5
E: evens
2
4
6
O.sync(E)
1,2
3,4
5,6


O: odds
1
3
5
E: evens
2
4
6
O.wait(E)
1
3

(Программирование)

(**RxJS** *(Microsoft ReactiveExtensions) github.com/Reactive-Extensions/RxJS)*

(**Bacon** *(Juha Paananen @raimohanska) github.com/baconjs/baconjs)*

(**Kefir** *(Roman Pominov @rpominov) github.com/pozadi/kefir)*

(**Warden.js** *(Trdat Mkrtchyan @zeffirsky) github.com/zefirka/Warden.js)*

(**ProAct** *(Nickolay Tzvetinov) github.com/proactjs/proactjs)*

Без FRP

```
var timer;

button.addEventListener('click', function(){
  clearTimeout(timer);
  setTimeout(function(){
    doSomething();
  }, 1000)
});
```

С FRP

```
Bacon.fromEventTarget(button, "click")
  .debounce(1000)
  .onValue(doSomething)
```

Без FRP

```
$window.on('scroll', function() {
  var scrollTop = $window.scrollTop(),
    newFlag = scrollTop >= scrollDist;

  if (flag != newFlag) {

    if (flag = newFlag) {
      target.addClass('b-header--fixed');
      animateHeaderControls();
    } else {
      target.removeClass('b-header--fixed');
      animateHeaderControls(false);
    }
  }
});
```

С FRP

```
function animateHeaderControls(hide){
  target[hide ? 'removeClass' : 'addClass']('b-header--fixed');
  ...
}

function moreThanScrollDist(){
  return $window.scrollTop() > scrollDist;
}

$window.asEventStream('scroll')
  .map(moreThanScrollDist)
  .onValue(animateHeaderControls)
```

Создание потоков

```
function getData(src){
  return Warden.makeStream(function(trigger){
    $.get(src, trigger);
  }).bus();
}
```

```
var api = {};
var stream = Warden.makeStream(function(trigger){
  this.getData = function(src){
      $.get(src, trigger);
      return stream;
  }
}, api).bus();

var getData = api.getData
```

Processing

```
getData('/api?query=users')
  .filter(function(response){
    return new Date() - response.changed.date > 1000 * 3600;
  })
  .map('.users')
  .listen(sideEffects);
```

(Middlewares)

FRP + Transducers

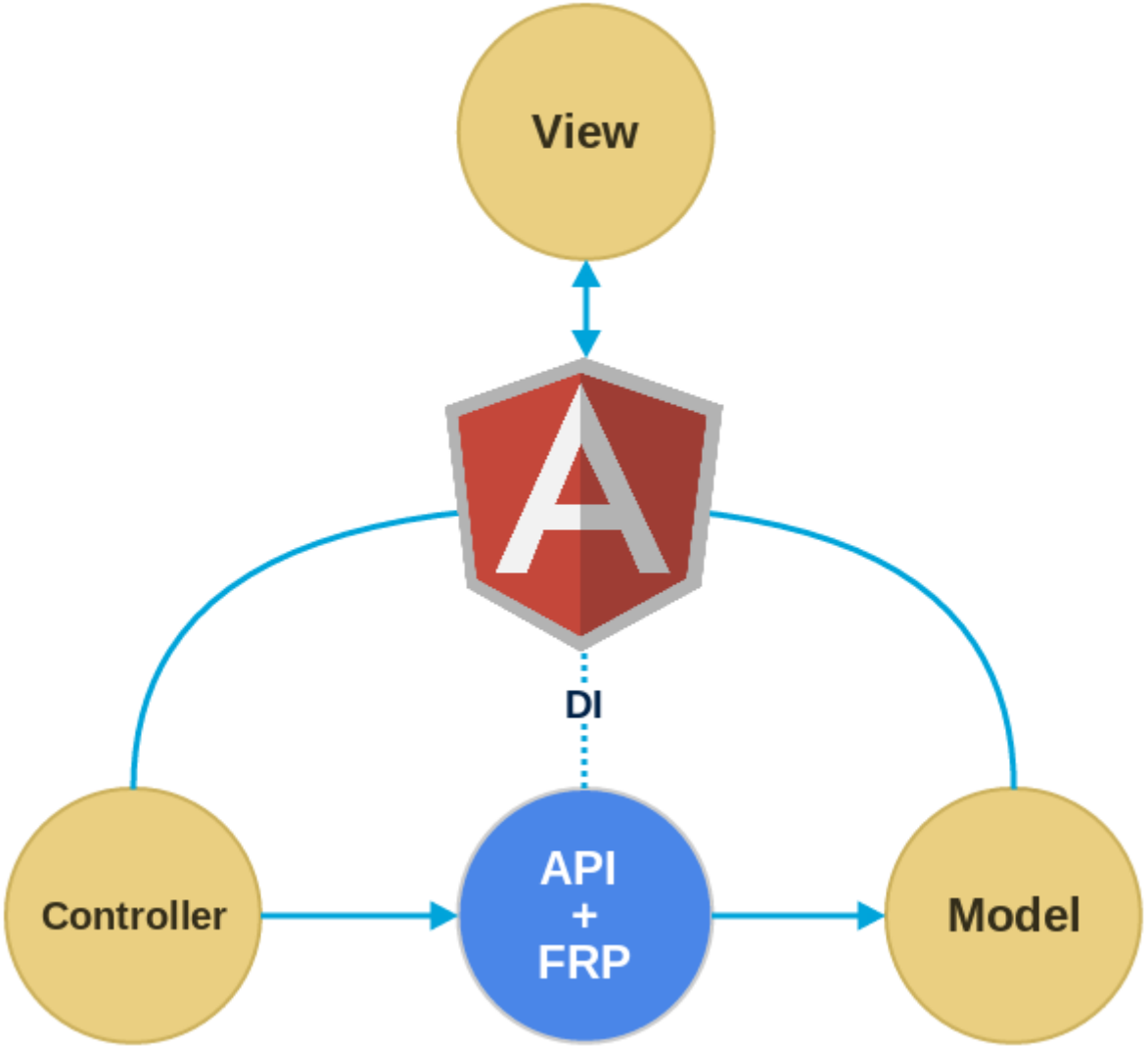
```
var transducer =  _t.compose(
  _t.filter(isActivePlayer),
  _t.pluck('score'),
  _t.map(add));

var summaryScores = getPlayersData(url).transduce(transducer);
```

FRP + Ramda

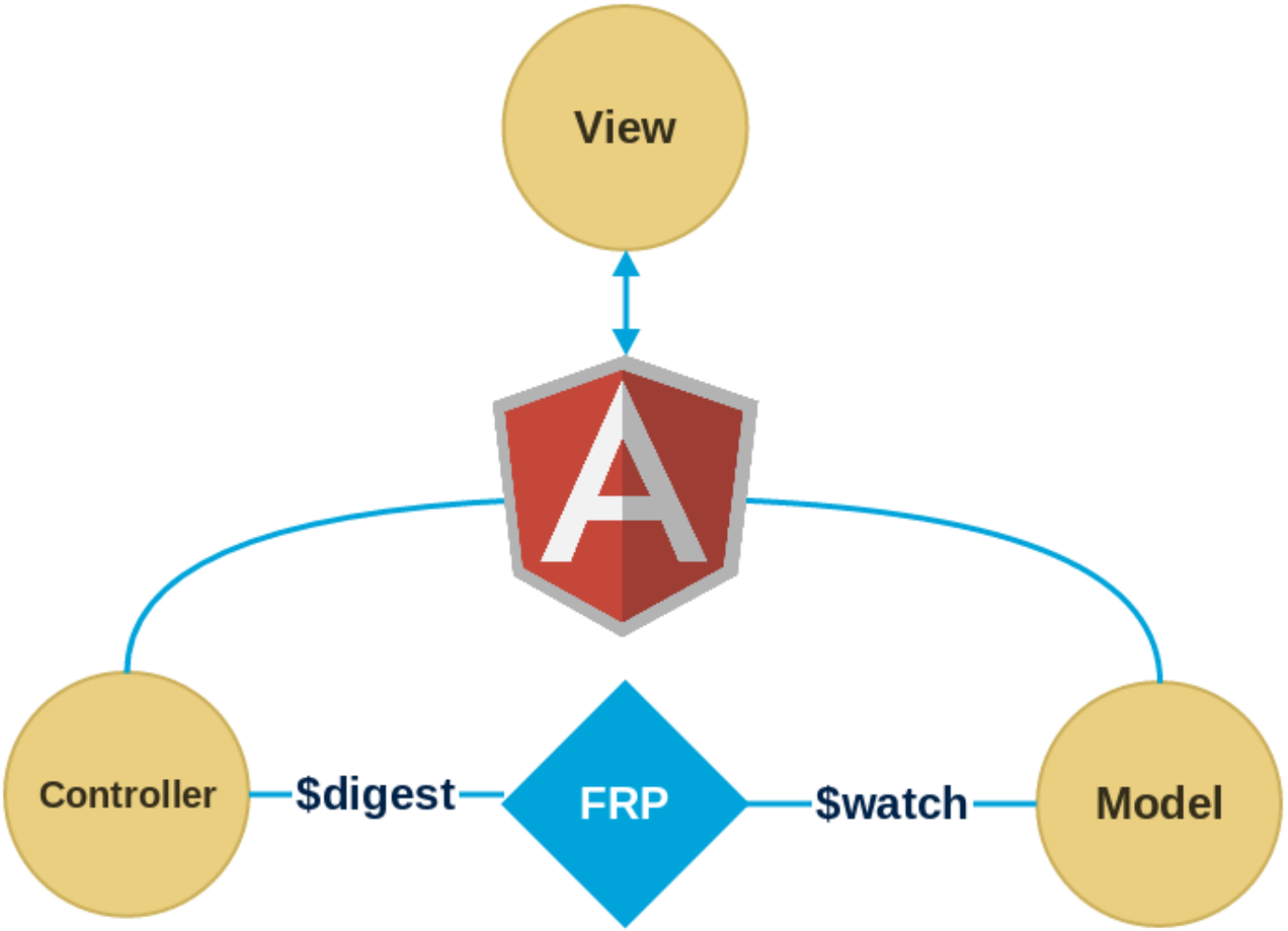
```
var summaryScores = getPlayerData(url)
  .map(R.map(R.add, R.pluck('score', R.filter(isPlayerActive))))
```

(Интеграция)

(Angular: случай 1)



(Angular: случай 2)



API

```
api.http = Warden.makeStream(function(emit){
  var $http = app.inject('$http');

  this.get = function(url, data){
    $http.get(url, data).then(emit);
    return api.http;
  }

}, api).bus();
```

Controller

```
function someCtrl($scope){
  $scope.users = [];

  api.get('api:users')
    .map('.users')
    .bindTo($scope, 'users')
}
```

View

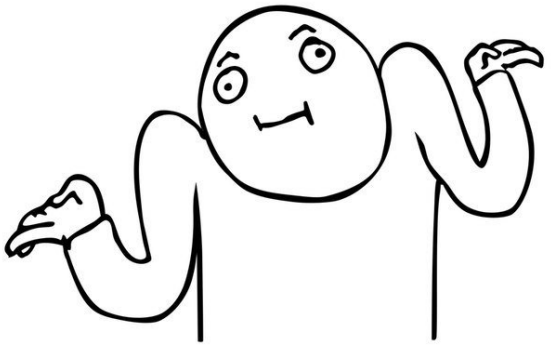
```
<ul>
  <li ng-repeat="user in users">
    ...
  </li>
</ul>
```

Controller

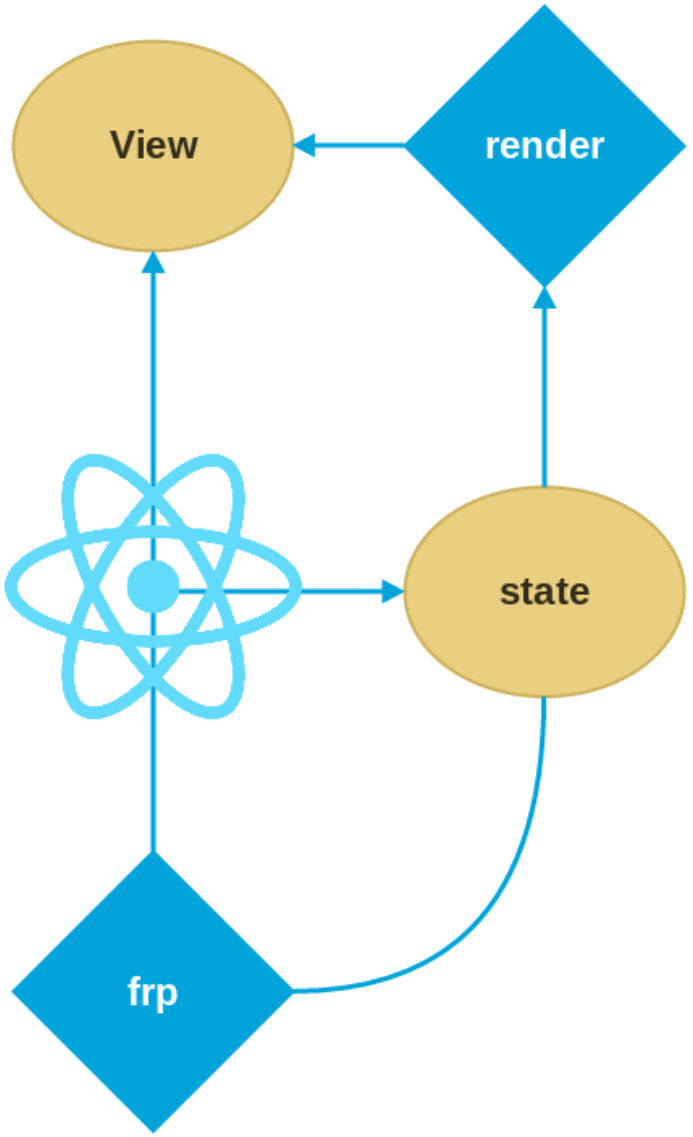
```
function someCtrl($scope){
  $scope.users = [];
  $scope.query = '';

  $scope.$stream('query')
    .debounce(500)
    .map('.newValue')
    .filter(notEmpty)
    .onValue(function(query){
      $http.get('/search', { query : query })
        .bindTo($scope, 'users');
    });
}
```

View

```
<input type='text' ng-model="query" value={{query}}>
<ul>
  <li ng-repeat="user in users">
    {{user}}
  </li>
</ul>
```



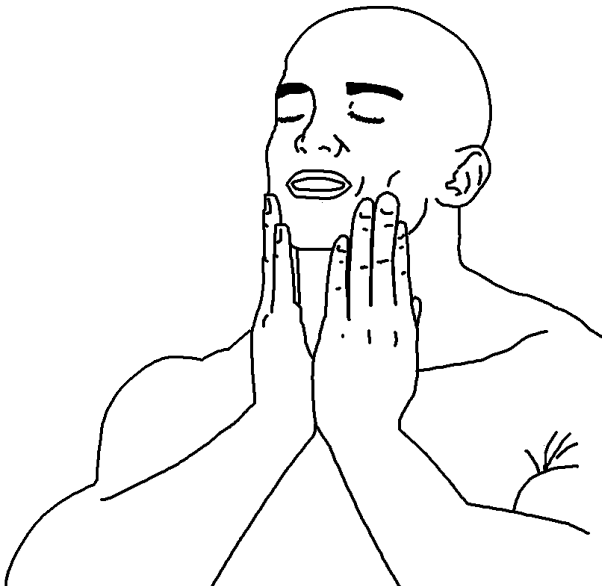
(React)



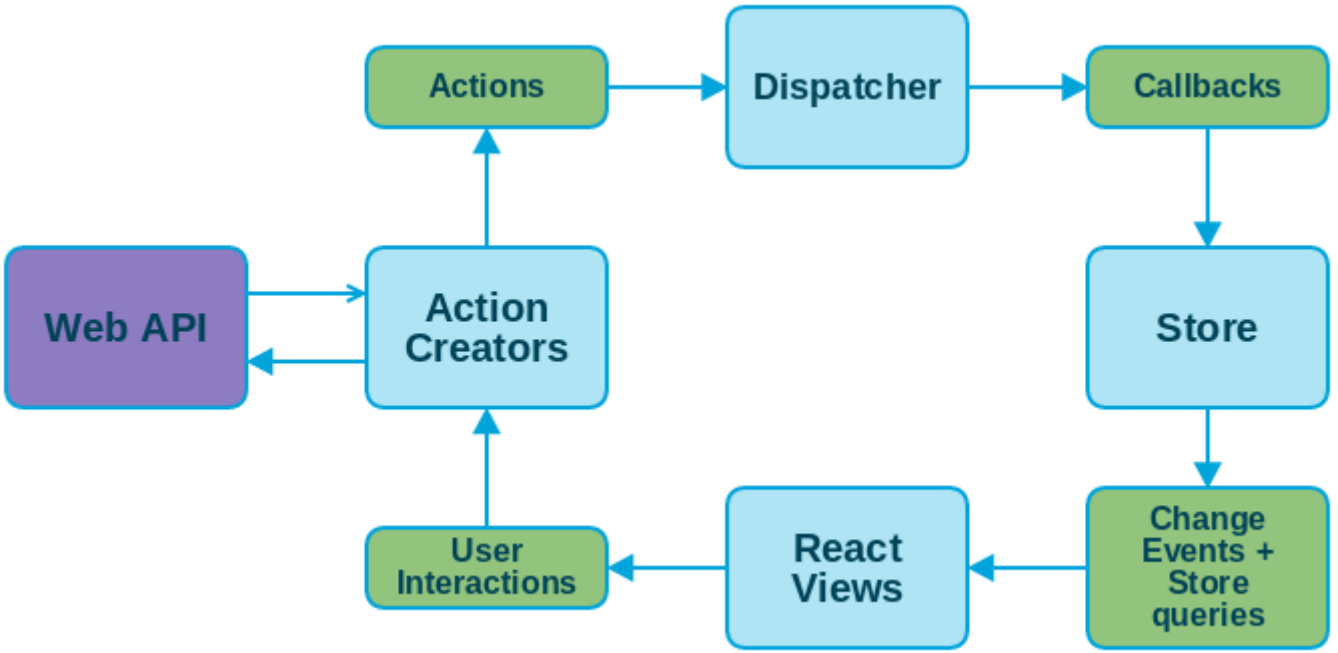
(Чуть больше магии,
но мы знаем как это работает)

(Отлично встраивается
во FLUX)

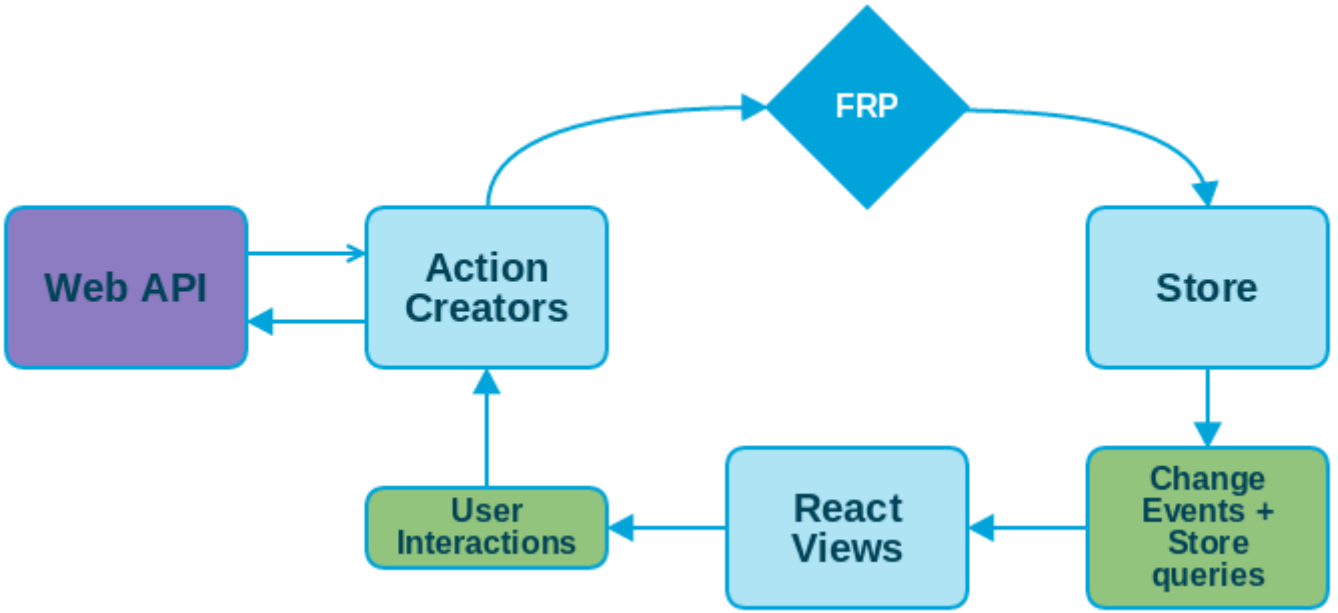
(Можно, наконец, понять,
что такое монада)



(FLUX)


Actions
Dispatcher
Callbacks
Web API
Action Creators
Store
User Interactions
React Views
Change Events + Store queries

(FLUX + FRP)


FRP
Web API
Action Creators
Store
User Interactions
React Views
Change Events + Store queries