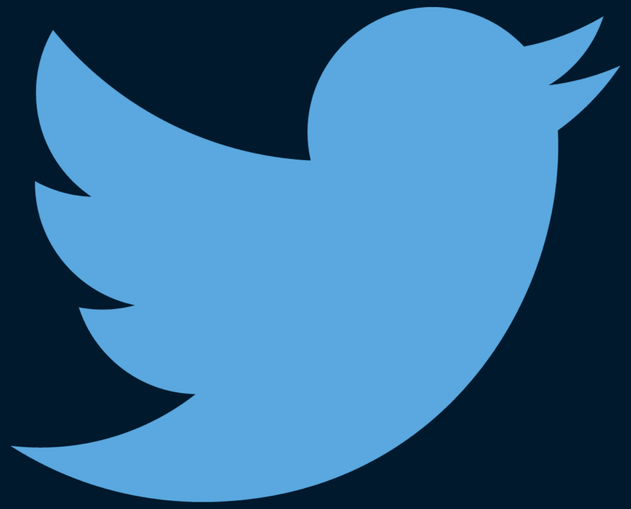
Simplifying the data layer

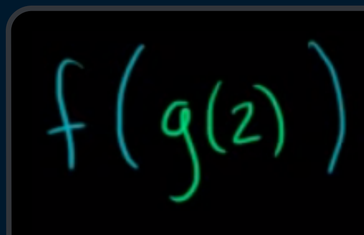
Kevin Robinson
@krob



fabric


answers
No analysis required.
Finally, mobile app analytics you don't need to analyze.

React!



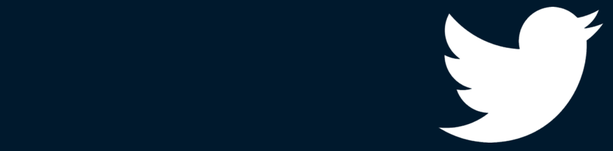
The data layer

Sharing information

Optimistic updates, offline

Real-time data, collaboration

Express this **simply** and **directly**

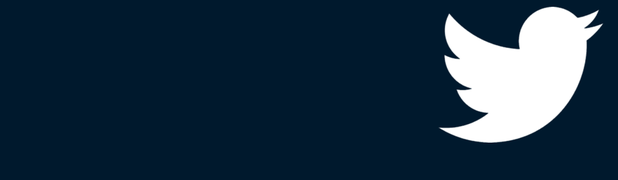


Simplifying

Collaborating across the stack

Push effects out to the edge

Embrace the log

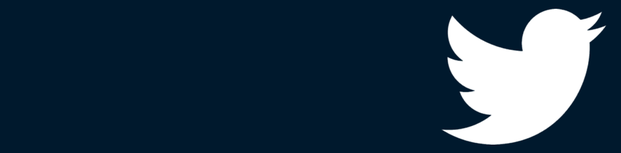


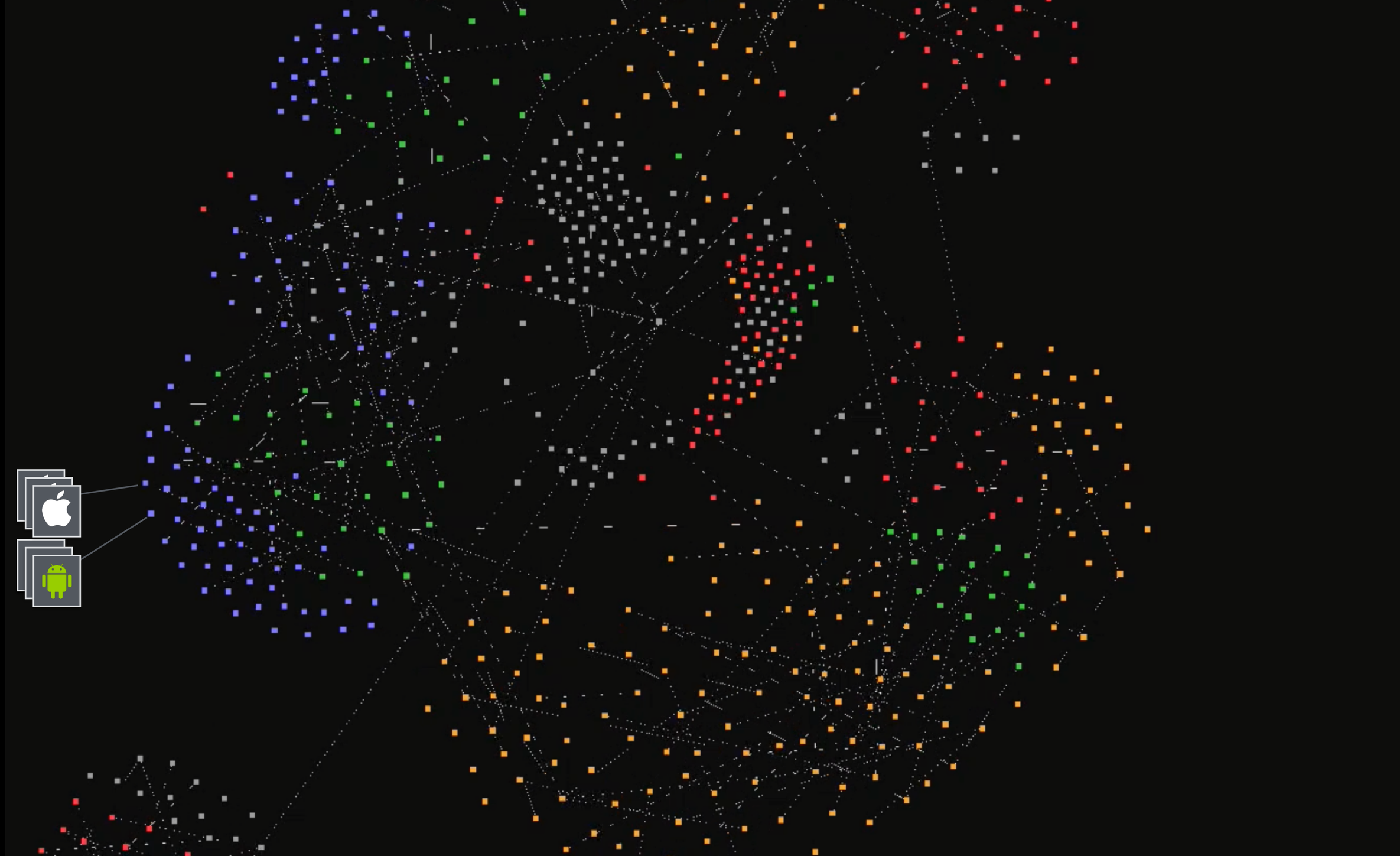
Simplifying

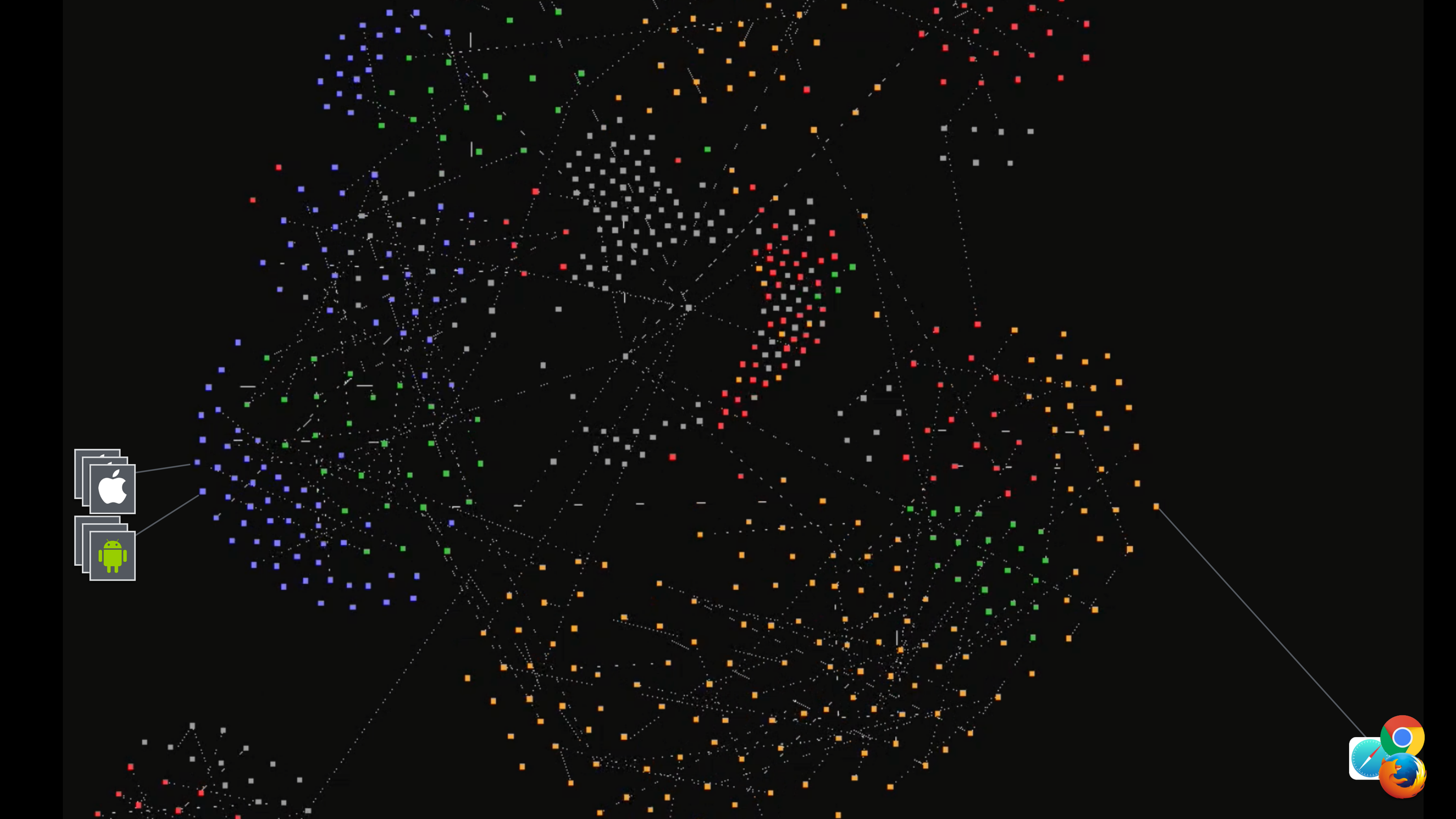
Collaborating across the stack

Push effects out to the edge

Embrace the log







Computers are for humans

Computers are for humans

Acknowledge latency

Events coming in at over 100M/min

Can’t push it all

Lambda architecture

Log, with different computed “views”

Optimized for product development

Fast, eventually consistent endpoints

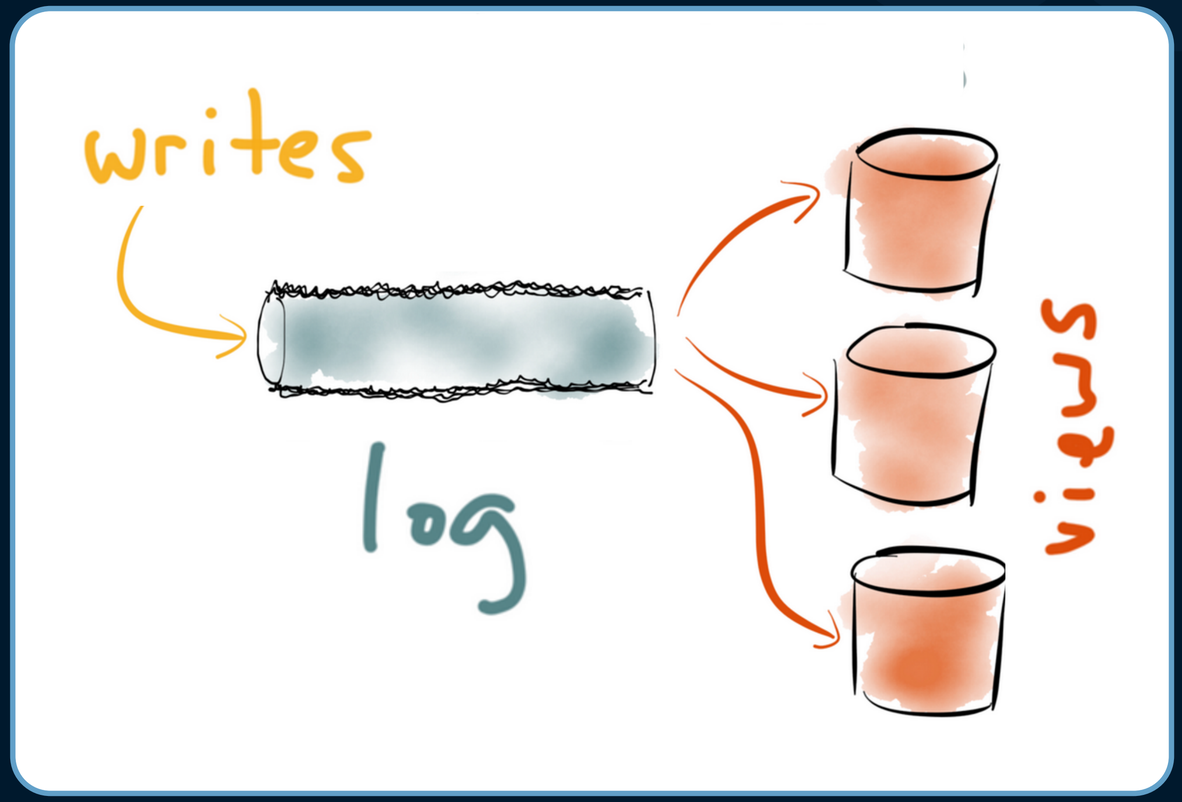
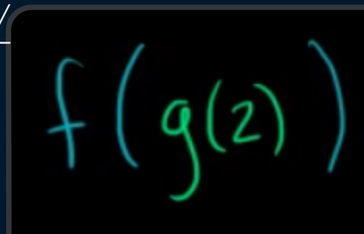


Diagram: http://blog.confluent.io/2015/03/04/

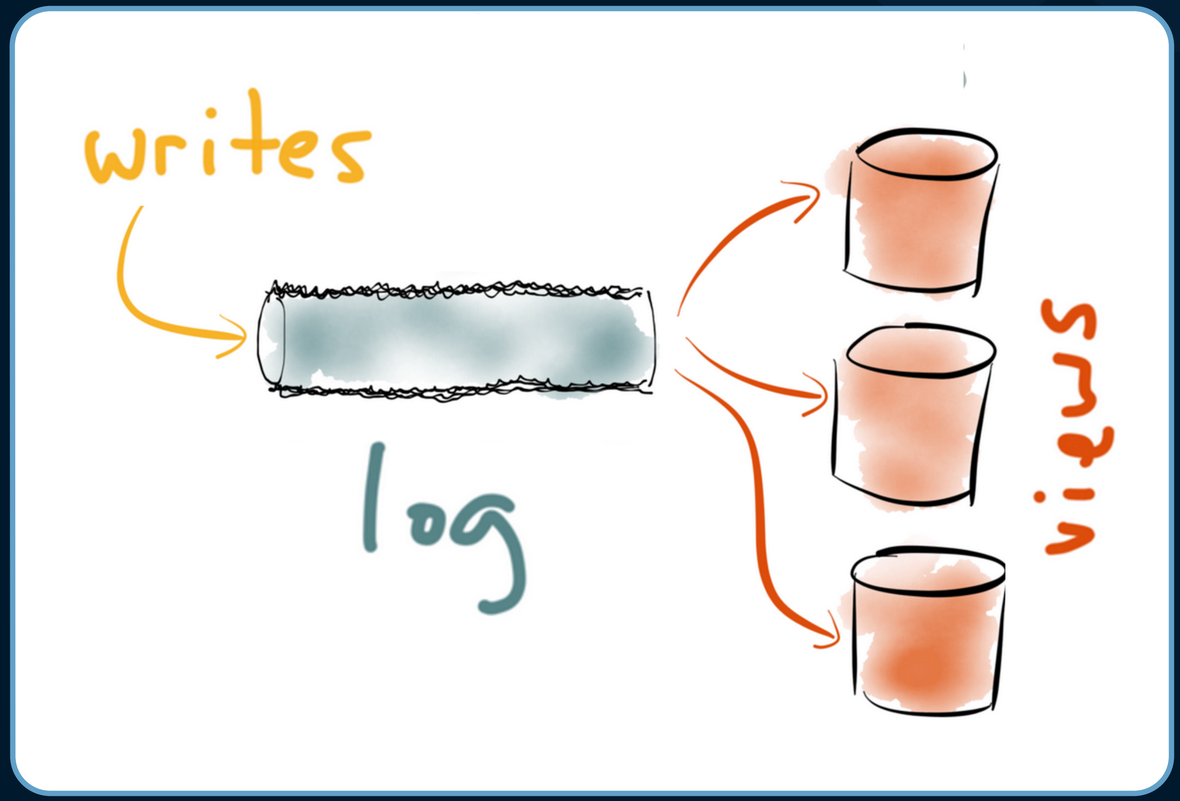


Lambda architecture

Log, with different computed “views”

Optimized for product development

Fast, eventually consistent endpoints



More details: https://blog.twitter.com/2015/handling-five-billion-sessions-a-day-in-real-time

Bound latency

Bound latency

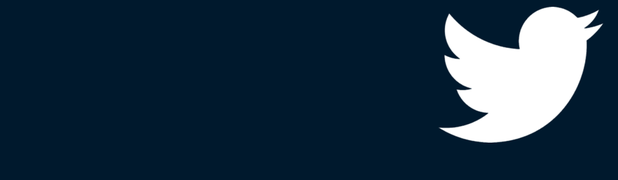
Bound latency

Simplifying

Collaborating across the stack

Push effects out to the edge

Embrace the log

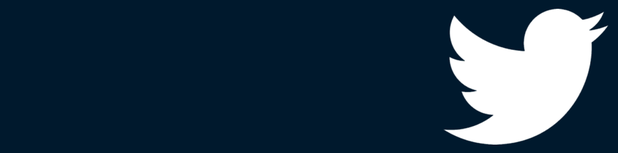


Simplifying

Collaborating across the stack

Push effects out to the edge

Embrace the log



How do we describe this?

Flux

How do we describe this?

USERS COMPONENT

```
ActiveUsersWidget = React.createClass
  getInitialState: ->
    activeUsers: null

  componentDidMount() {
    this.activeUsersProcess = new ActiveUsersProcess({
      appId: this.props.appId,
      timeRange: this.props.timeRange,
      onDataReceived: this.onActiveUsersData
    });
    this.activeUsersProcess.start();
  }

  componentWillUnmount() {
    this.activeUsersProcess.close();
  }

  onActiveUsersData(activeUsers) {
    this.setState({ activeUsers });
  }

  render() {
```

How do we describe this?

USERS COMPONENT

```
ActiveUsersWidget = React.createClass
  getInitialState: ->
    activeUsers: null

  componentDidMount() {
    this.activeUsersProcess = new ActiveUsersProcess({
      appId: this.props.appId,
      timeRange: this.props.timeRange,
      onDataReceived: this.onActiveUsersData
    });
    this.activeUsersProcess.start();
  }

  componentWillUnmount() {
    this.activeUsersProcess.close();
  }

  onActiveUsersData(activeUsers) {
    this.setState({ activeUsers });
  }

  render() {
```

ACTIVE USERS PROCESS

Sharing across components

USERS COMPONENT

```
ActiveUsersWidget = React.createClass
  getInitialState: ->
    activeUsers: null

  componentDidMount() {
    this.activeUsersProcess = new ActiveUsersProcess({
      appId: this.props.appId,
      timeRange: this.props.timeRange,
      onDataReceived: this.onActiveUsersData
```

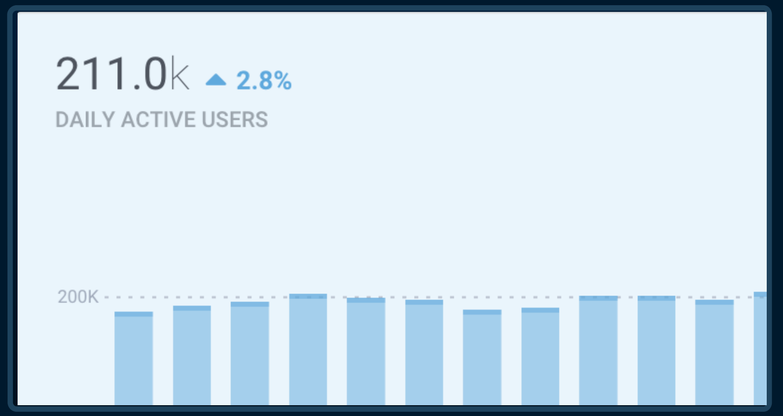
ACTIVE USERS PROCESS

Sharing across components

USERS COMPONENT

```
ActiveUsersWidget = React.createClass
  getInitialState: ->
    activeUsers: null

  componentDidMount() {
    this.activeUsersProcess = new ActiveUsersProcess({
      appId: this.props.appId,
      timeRange: this.props.timeRange,
      onDataReceived: this.onActiveUsersData
```

ACTIVE USERS PROCESS



Sharing across components

USERS COMPONENT

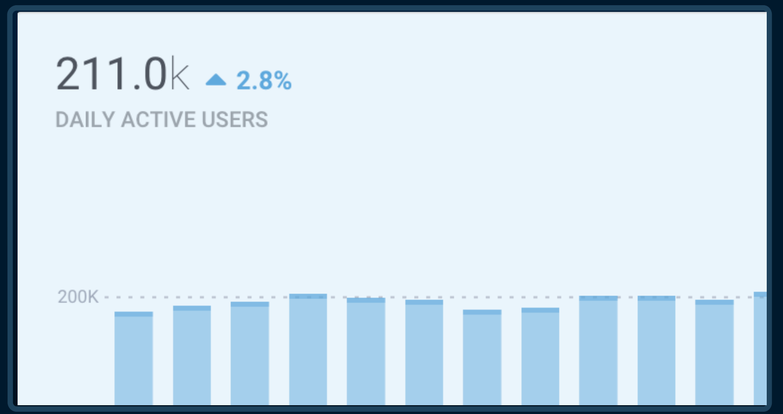
```
ActiveUsersWidget = React.createClass
  getInitialState: ->
    activeUsers: null

  componentDidMount() {
    this.activeUsersProcess = new ActiveUsersProcess({
      appId: this.props.appId,
      timeRange: this.props.timeRange,
      onDataReceived: this.onActiveUsersData
```

ACTIVE USERS PROCESS

EXPANDED USERS COMPONENT

```
ActiveUsersWidget = React.createClass
  getInitialState: ->
    activeUsers: null

  componentDidMount() {
    this.activeUsersProcess = new ActiveUsersProcess({
      appId: this.props.appId
```

ACTIVE USERS PROCESS



Joseph Savona
@en_JS

Following

The solution to async UI development is not async APIs: declarative, functional approaches are more powerful and easier to reason about.

RETWEETS
9

FAVORITES
16

4:57 PM - 11 Jun 2015

Push effects to the edge

DESCRIPTION OF DOM

```
render() {
  const latestValue = _.last(this.state.activeUsers).userCount;
  return (
    <AnswersWidget>
      <Heading
        title="Active users right now" value={latestValue} />
      <LogarithmicChart timeseries=this.state.{activeUsers} />
    </AnswersWidget>
  );
}
```

Push effects to the edge

DESCRIPTION OF PROCESSES

```
// This is what I want - components describe what
// processes they need as a function of props and state.
declareNeeds(props, state)  {
  return {
    activeUsers:
      processKey: ActiveUsersProceess.key
      params:
        appId: props.appId
        timeRange: props.timeRange
  };
}
```

Write declaratively

DESCRIPTION OF DOM

```
render() {
  const latestValue = _.last(this.state.activeUsers).userCount;
  return (
    <AnswersWidget>
      <Heading
        title="Active users right now" value={latestValue} />
      <LogarithmicChart timeseries=this.state.{activeUsers} />
    </AnswersWidget>
  );
}
```

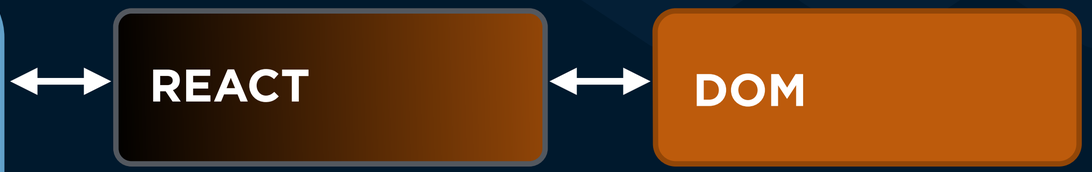
Write declaratively

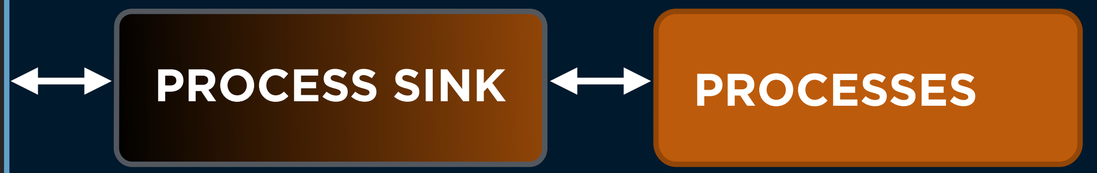
DESCRIPTION OF DOM

```
render() {
  const latestValue = _.last(this.state.activeUsers).userCount;
  return (
    <AnswersWidget>
      <Heading
        title="Active users right now" value={latestValue} />
      <LogarithmicChart timeseries=this.state.{activeUsers} />
    </AnswersWidget>
  );
}
```

DESCRIPTION OF PROCESSES

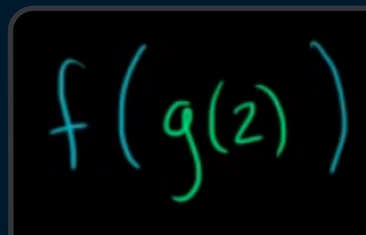
```
declareNeeds(props, state)  {
  return {
    activeUsers:
      processKey: ActiveUsersProceess.key
      params:
        appId: this.props.appId
        timeRange: this.props.timeRange
  };
```





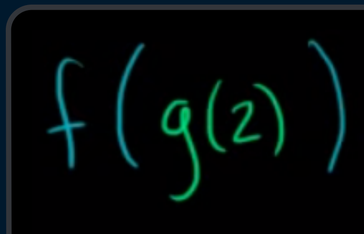
Functions are easy to factor

```
declareNeeds(props, state) {
  return {
    processKey: TopEventsProcess.key,
    params: {
      timeRange: props.timeRange
    }
  };
}
```

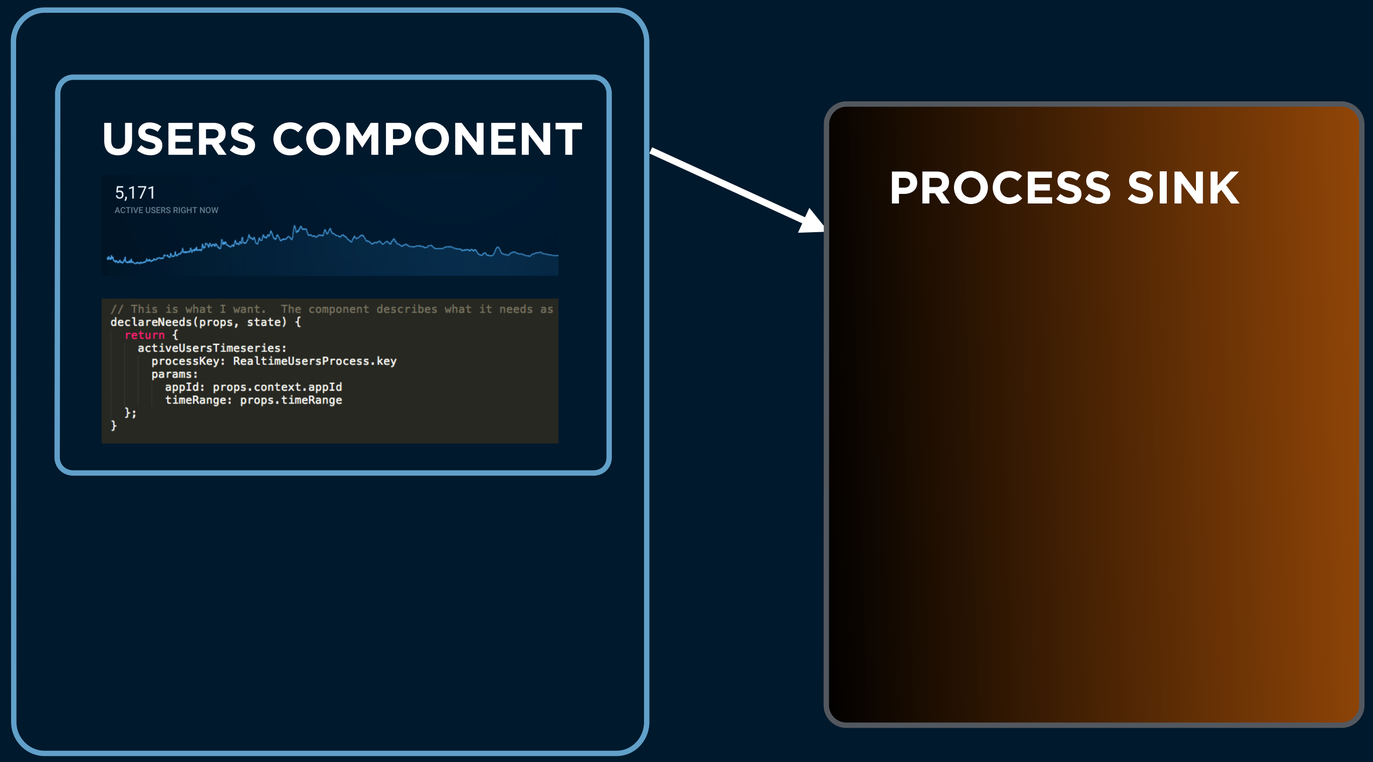


Functions are easy to factor

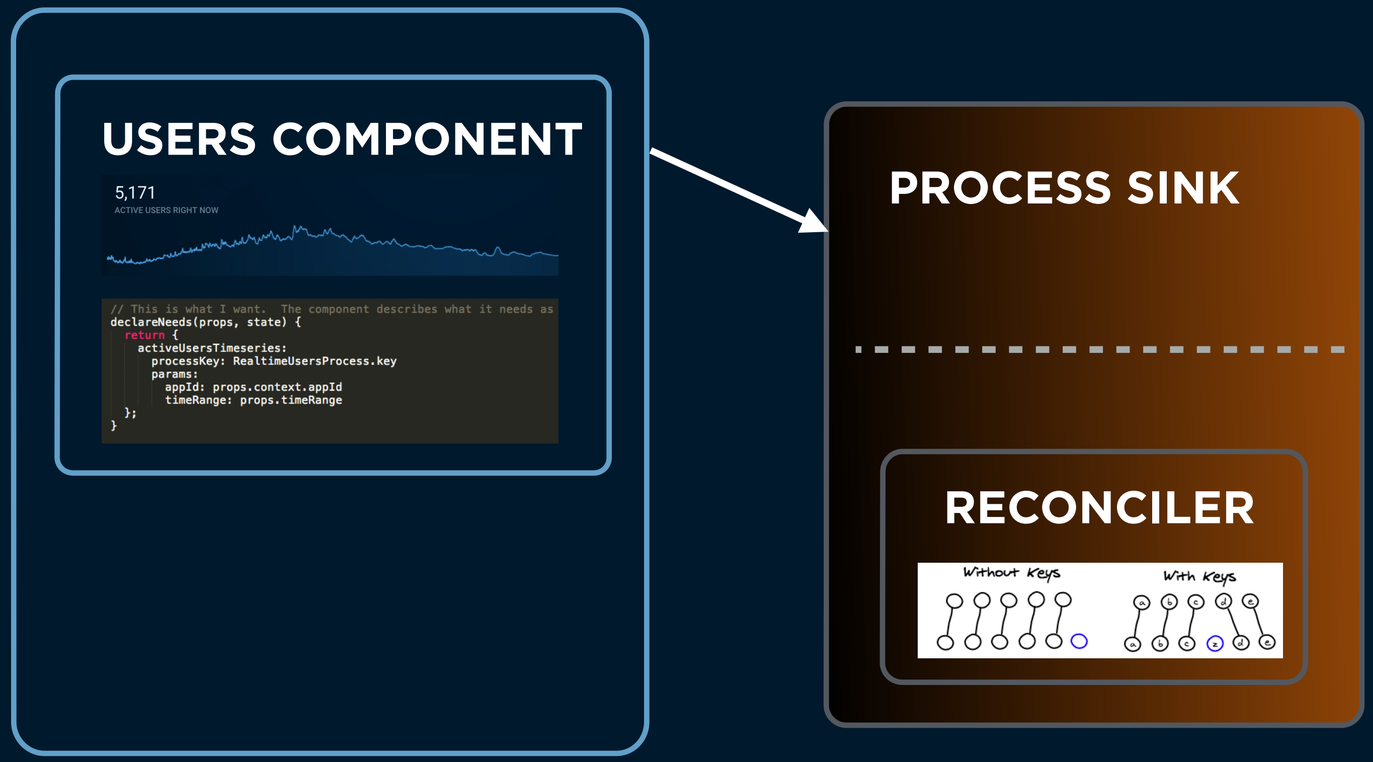
```
declareNeeds(props, state) {
  const topEventNeed = {
    processKey: TopEventsProcess.key,
    params: {
      timeRange: props.timeRange
    }
  };

  const topEventType = this.topEventType(state);
  const eventTimeseriesNeed = !topEventType ? {} : {
    processKey: EventTimeseriesProcess.key,
    params: {
      eventType: topEventType,
      timeRange: props.timeRange
    }
  };

  return Object.assign({}, topEventNeed, eventTimeseriesNeed);
}
```



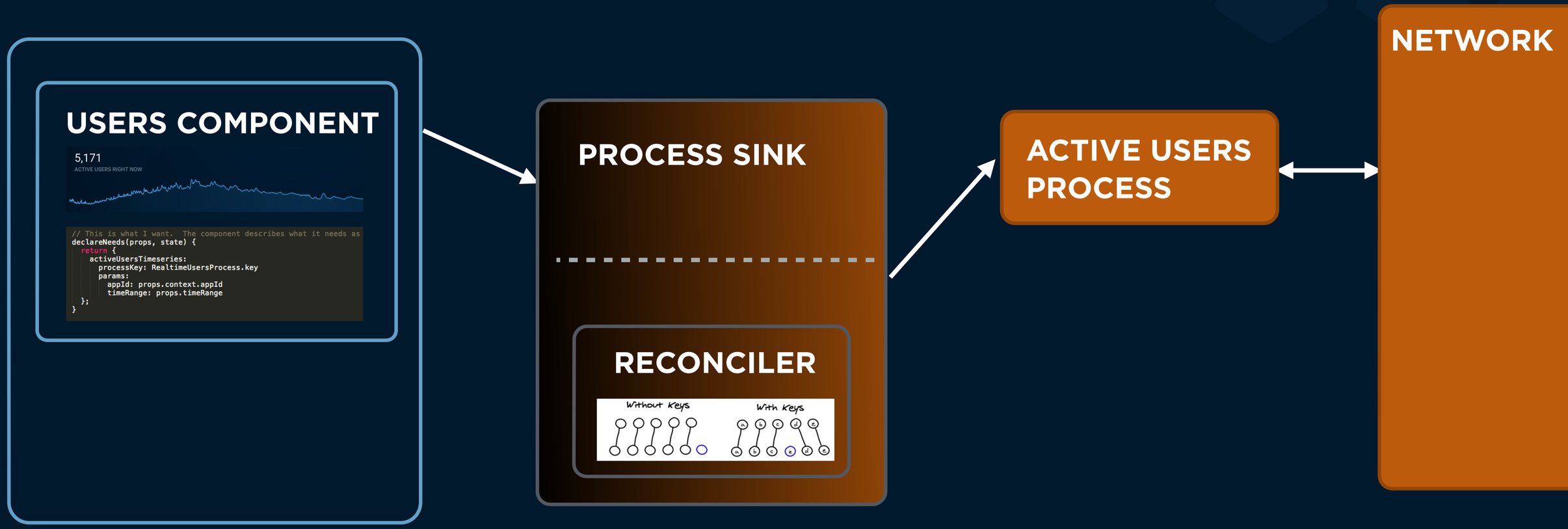
Components declare needs



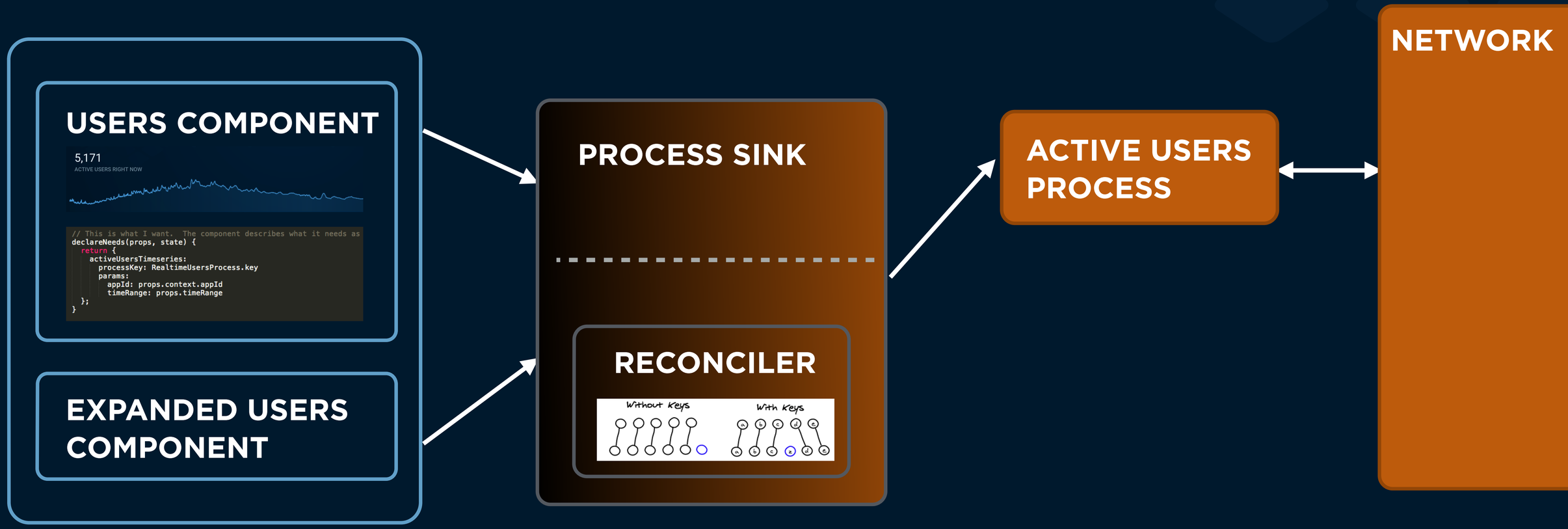
Library code interprets



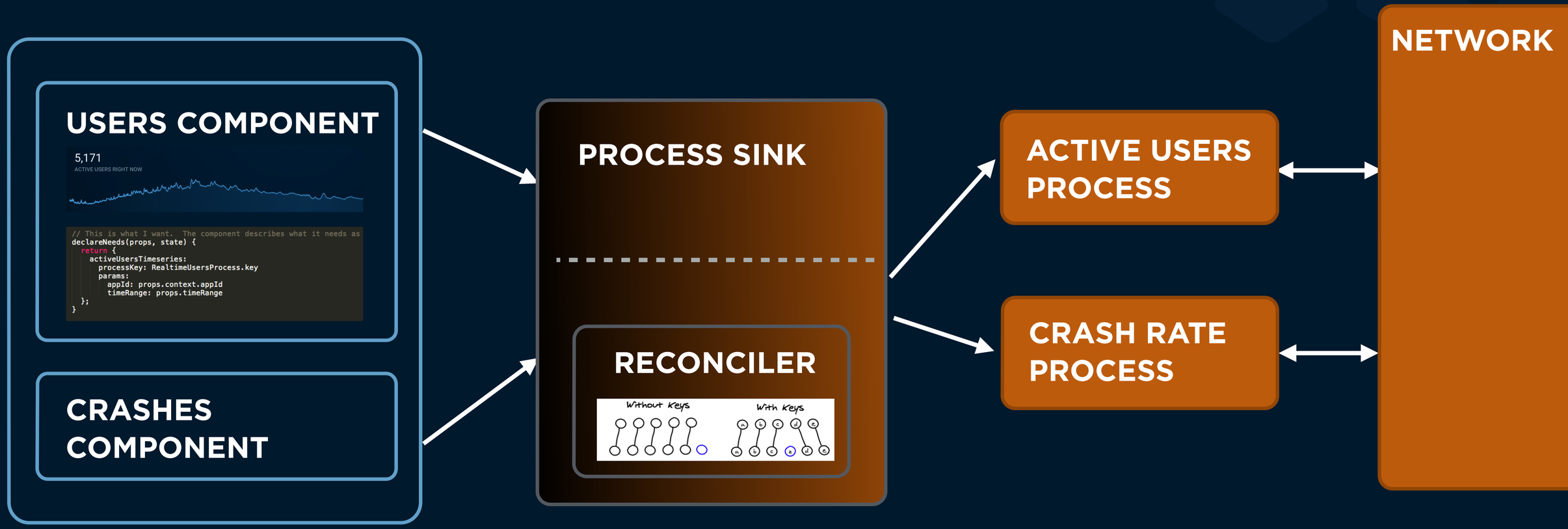
Library code reconciles



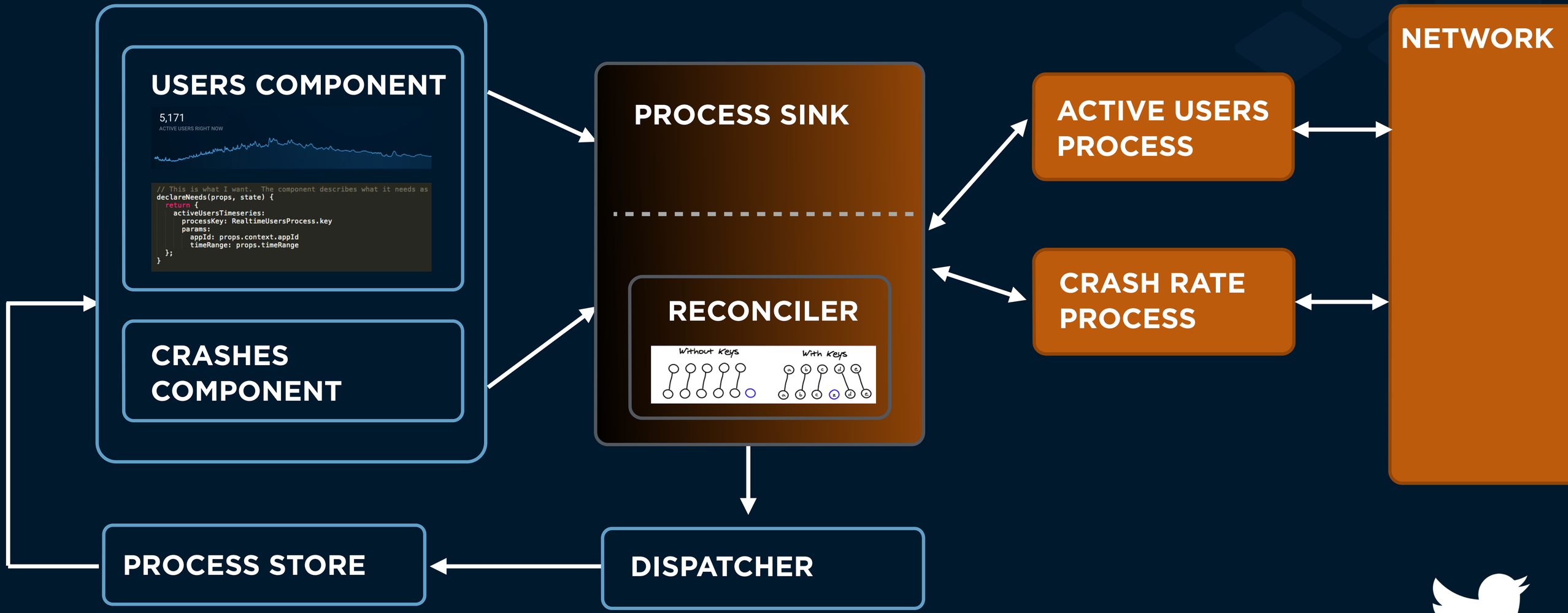
Share across components



Spin up processes as needed



Make data observable



Other applications

Server rendering

React Nexus

Composition

GraphQL, **Datalog**

Optimize requests

Relay, Om.next

Request and response

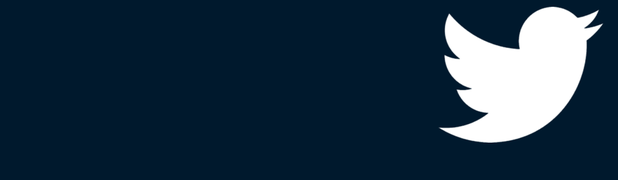
Elm

Simplifying

Collaborating across the stack

Push effects out to the edge

Embrace the log

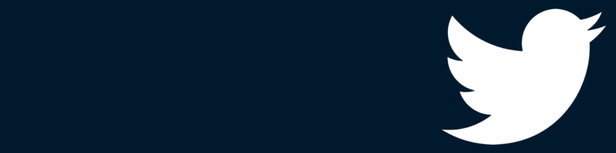


Simplifying

Collaborating across the stack

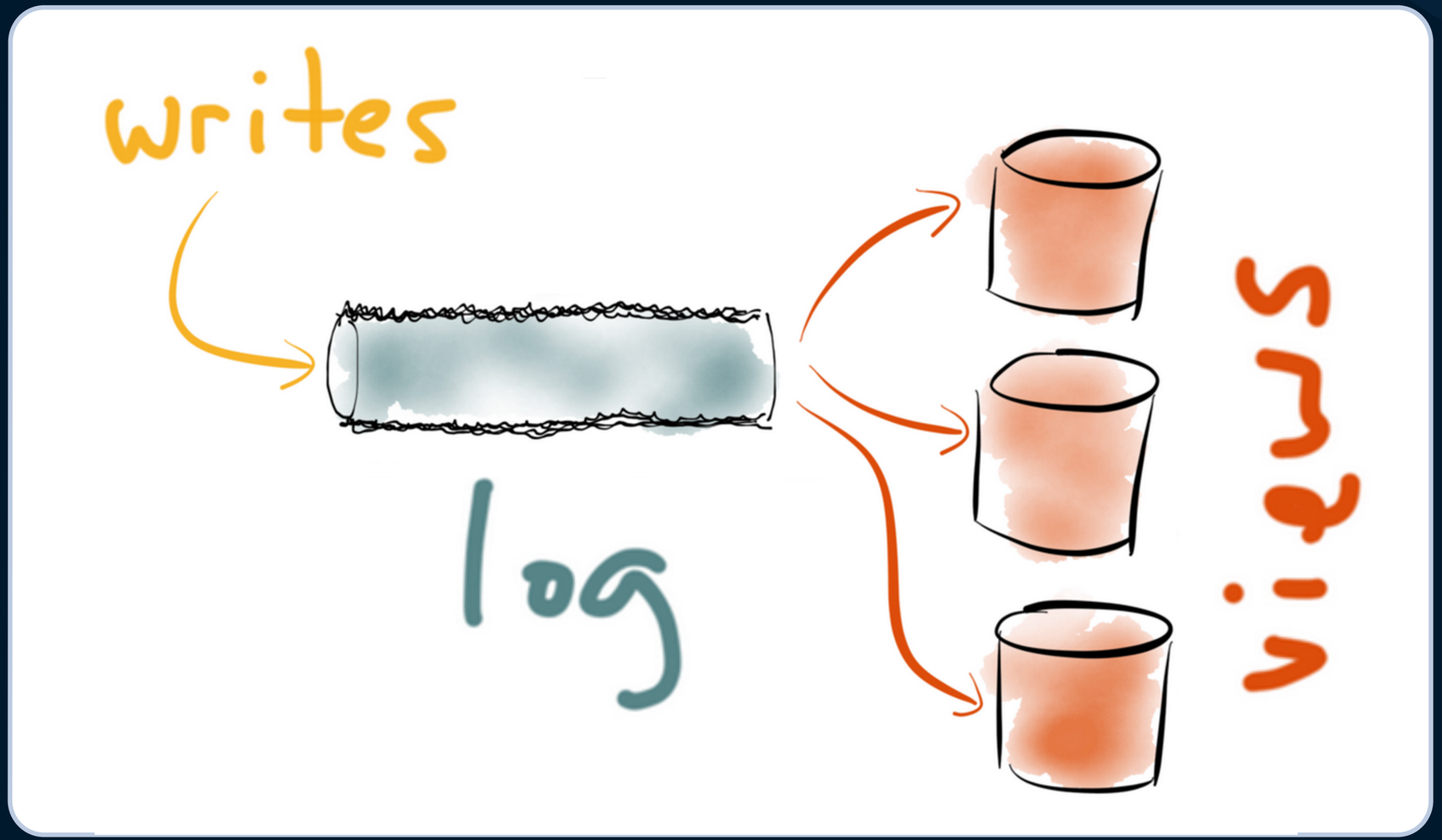
Push effects out to the edge

Embrace the log



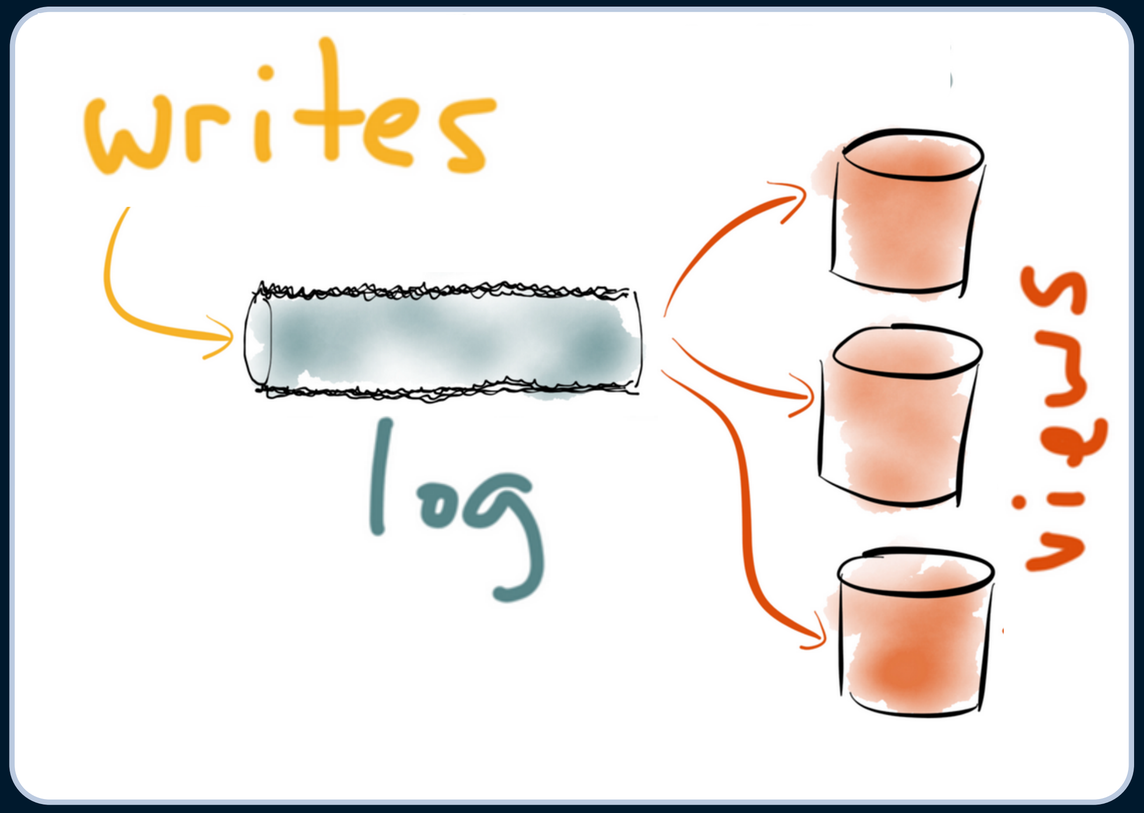
What?

Embrace the log



http://blog.confluent.io/2015/03/04/

Embrace the log

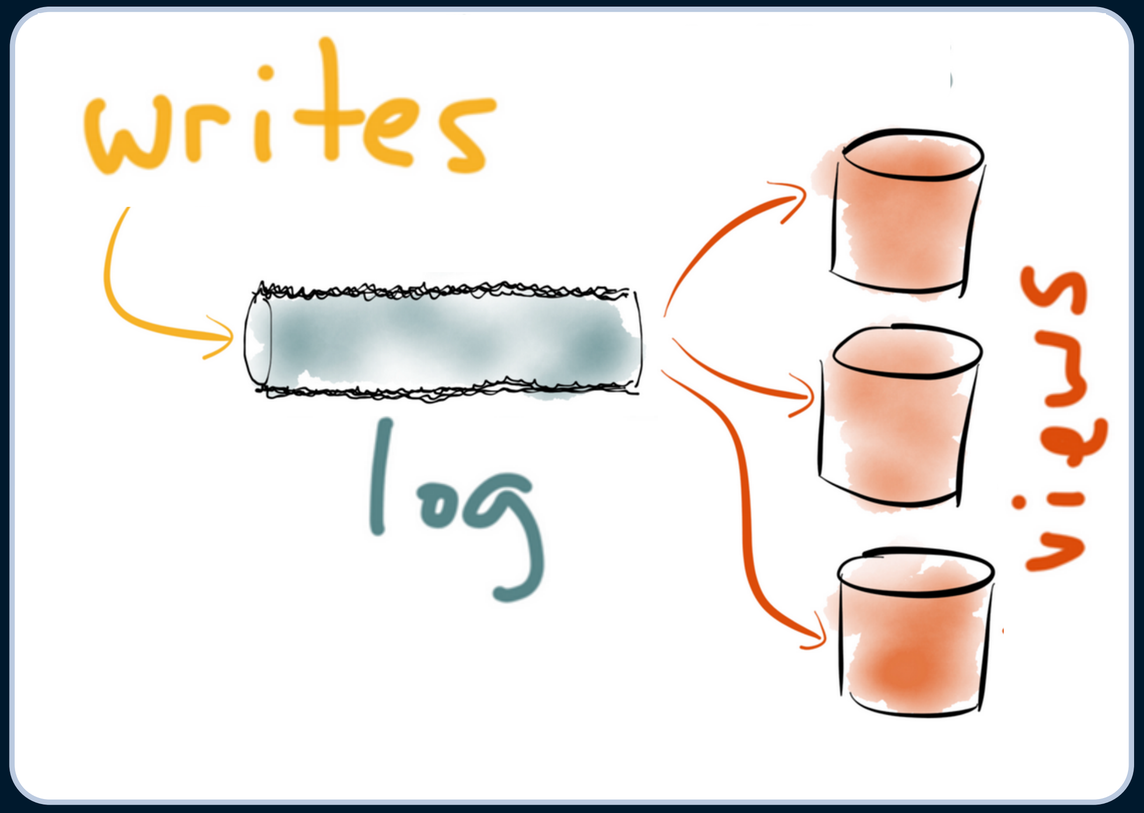


Writes:

~~mutate state in place~~

append-only stream of immutable facts

Embrace the log



Strengths

Isolation

Swapping computation

“Delay design”

Why?

Development superpowers

Hot reloading

Debugging

Recording user sessions

https://github.com/gaearon/redux

Data layer superpowers

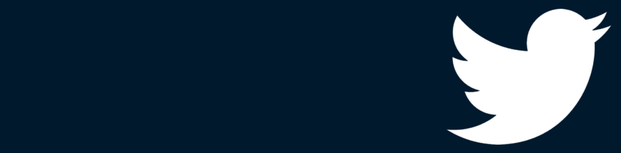
```
(aset js/window "undo"
  (fn [e]
    (when (> (count @app-history) 1)
      (swap! app-history pop)
      (reset! app-state (last @app-history)))))
```

https://github.com/omcljs/om

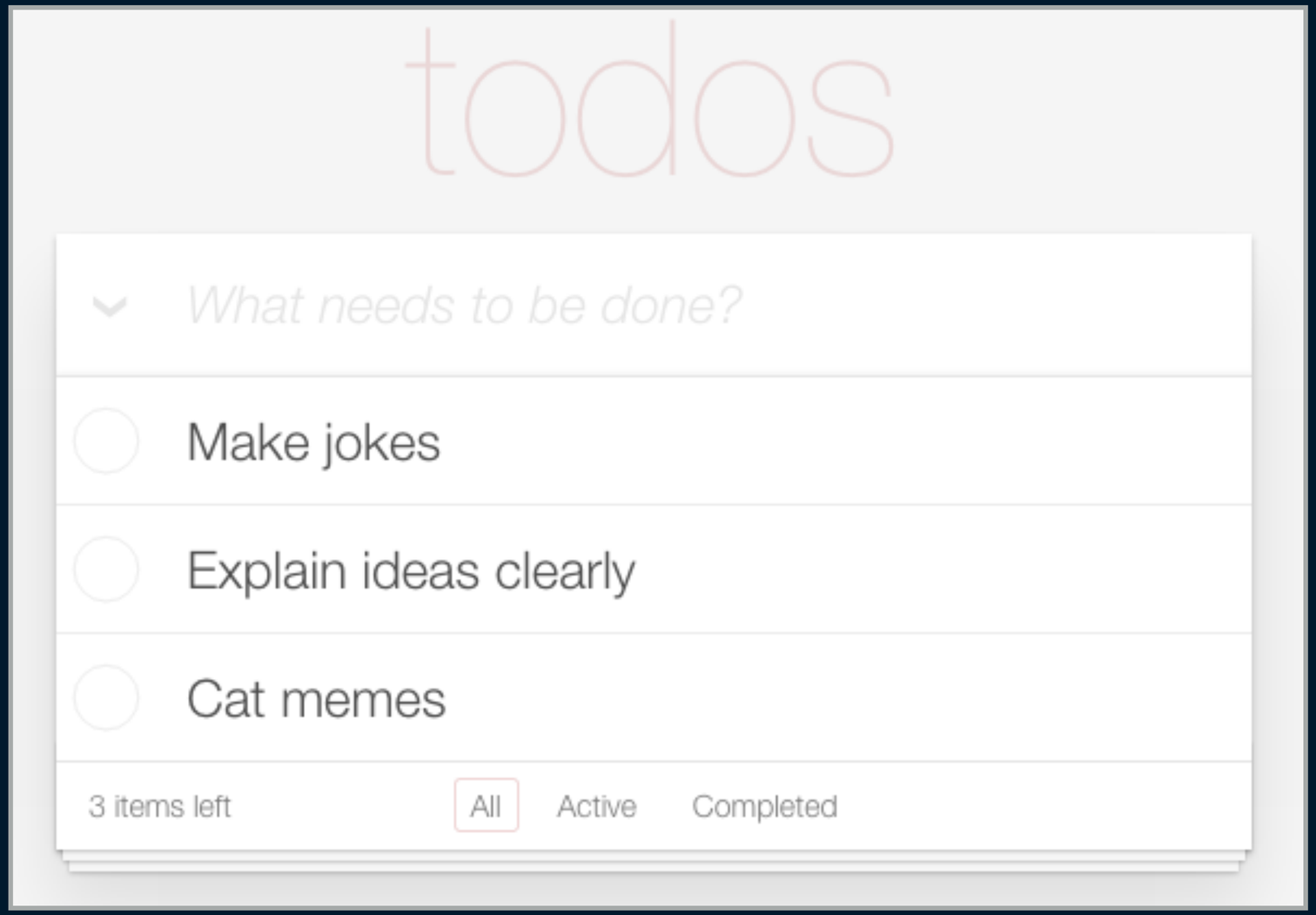
Real-time updates

Collaboration

Partial connectivity



Data layer superpowers



Embrace the log

```
timestamp  key                      payload
-------------------------------------------------------------------------------
 3000      USER_ADDED_TODO          {text: "Make jokes"}
 3005      CREATE_TODO_REQUEST      {request: {text: "Make jokes"}}
 3500      CREATE_TODO_RESPONSE     {request: {text: "Make jokes"}}, response: {id: 14}}
 6000      USER_ADDED_TODO          {text: "Explain ideas clearly"}
 8000      USER_ADDED_TODO          {text: "Cat memes"}
 ...
12000      TODOS_LIST_RESPONSE      [{id: 14, text: "Make funny jokes"}]
```

Embrace the log

```
timestamp   key                       payload
-------------------------------------------------------------------------------
 3000       USER_ADDED_TODO           {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```

Embrace the log

```
timestamp   key                       payload
-------------------------------------------------------------------------------
 3000       USER_ADDED_TODO           {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```


```
[ {id: 14, text: "Make jokes funny"} ]
```

Embrace the log

```
timestamp   key                       payload
-------------------------------------------------------------------------------
 3000       USER_ADDED_TODO           {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```

Embrace the log

```
timestamp   key                        payload
-------------------------------------------------------------------------------
 3000       USER_ADDED_TODO            {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST        {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE       {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO            {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO            {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE        [{id: 14, text: "Make funny jokes"}]
```


```
[ {id: 14, text: "Make jokes funny"},
  {text: "Explain jokes clearly"},
  {text: "Cat memes"} ]
```

Embrace the log

```
timestamp  key                       payload
-------------------------------------------------------------------------------
 3000      USER_ADDED_TODO           {text: "Make jokes"}
 3005      CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500      CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000      USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000      USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000      TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```

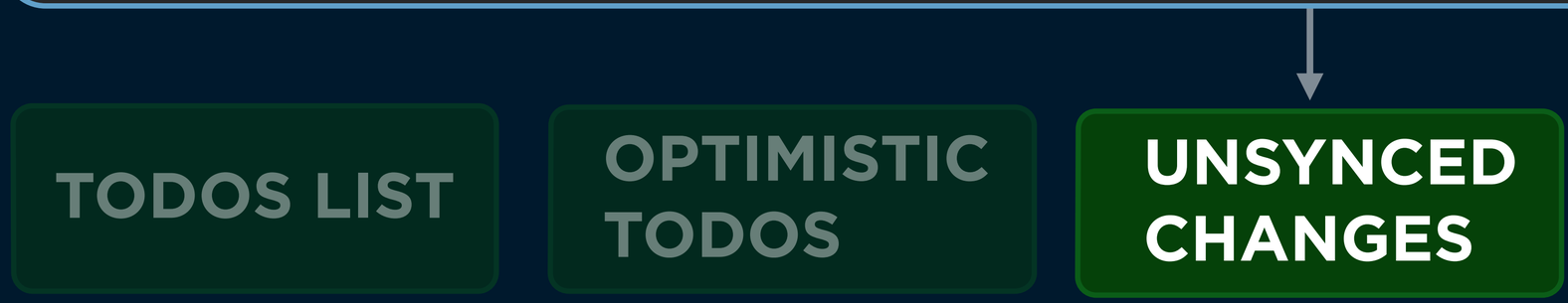
TODOS LIST

OPTIMISTIC TODOS

UNSYNCED CHANGES

Embrace the log

```
timestamp   key                       payload
-------------------------------------------------------------------------------
 3000       USER_ADDED_TODO           {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```




```
[ {text: "Explain jokes clearly"},
  {text: "Cat memes"} ]
```

Embrace the log

```
timestamp   key                       payload
-------------------------------------------------------------------------------
 3000       USER_ADDED_TODO           {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```

TODOS LIST

OPTIMISTIC TODOS

UNSYNCED CHANGES

HIGHLIGHT CONFLICTS

Embrace the log

```
timestamp   key                       payload
-------------------------------------------------------------------------
 3000       USER_ADDED_TODO           {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```

TODOS LIST

OPTIMISTIC TODOS

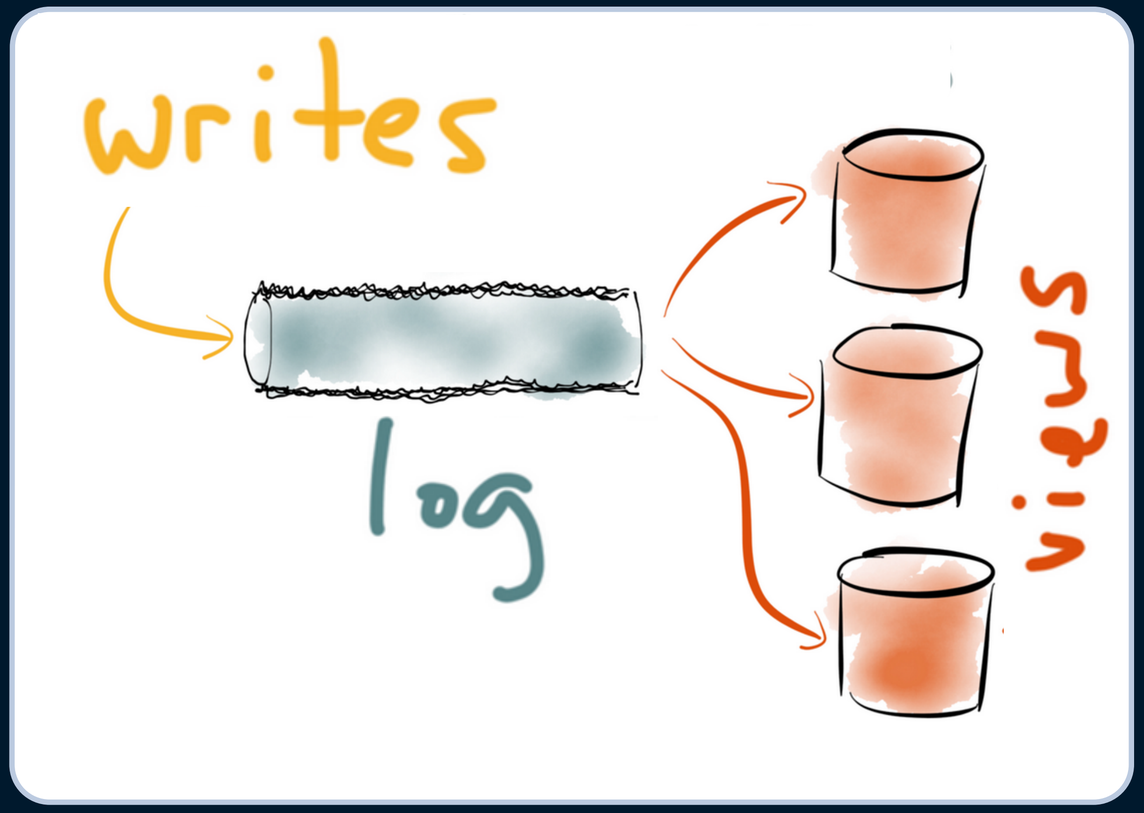
UNSYNCED CHANGES

HIGHLIGHT CONFLICTS

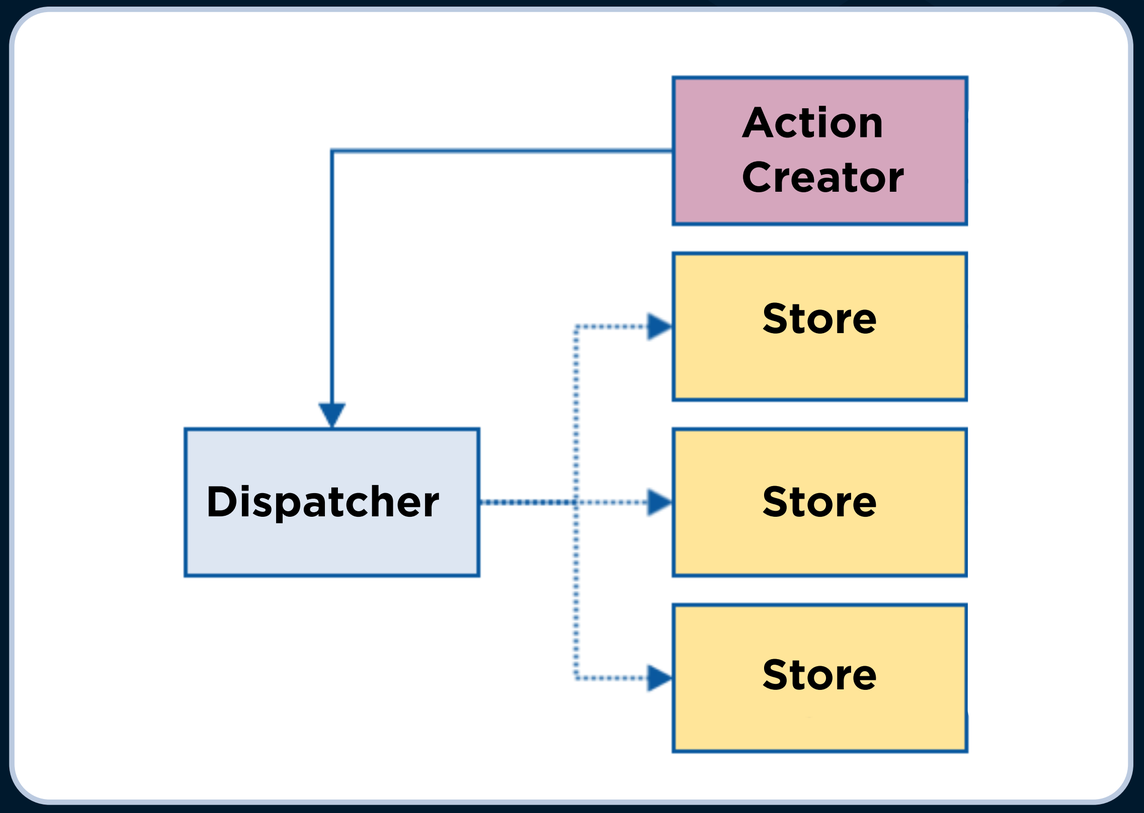
```
[ {local: {text: "Make jokes"},
   server: {id: 14, text: "Make jokes funny"}} ]
```

How?

Already partway there!



http://blog.confluent.io/2015/03/04/



http://fluxxor.com/what-is-flux.html

Already partway there!

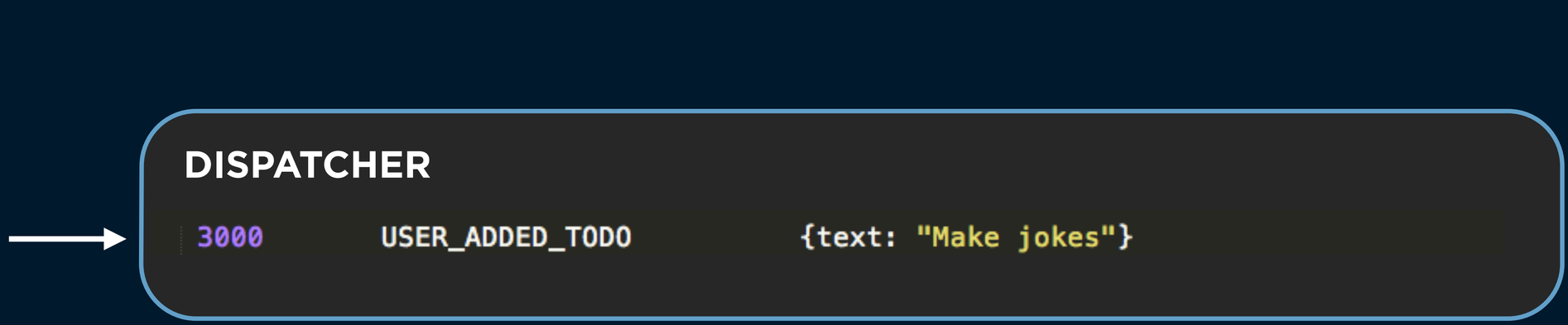
DISPATCHER

STORE

STORE

STORE

Already partway there!

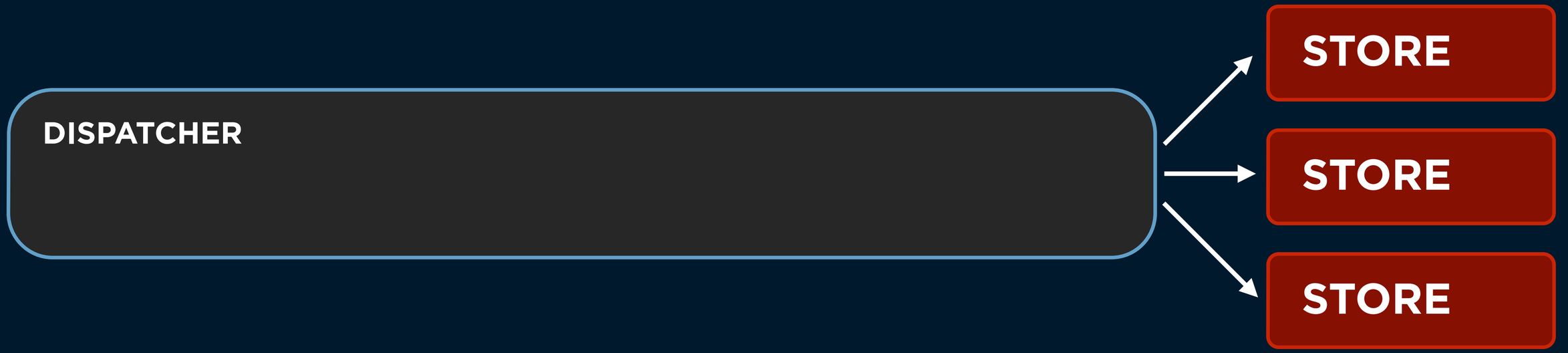


STORE

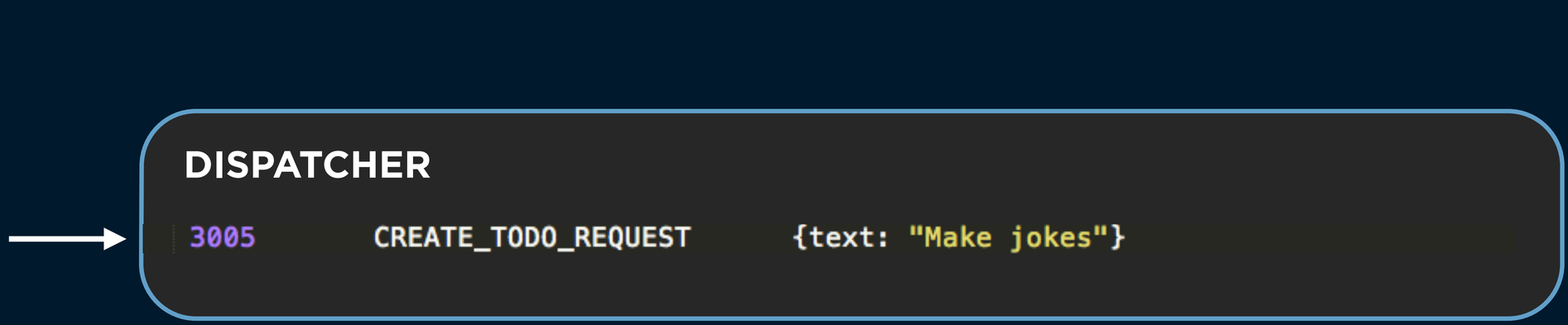
STORE

STORE

Already partway there!



Already partway there!

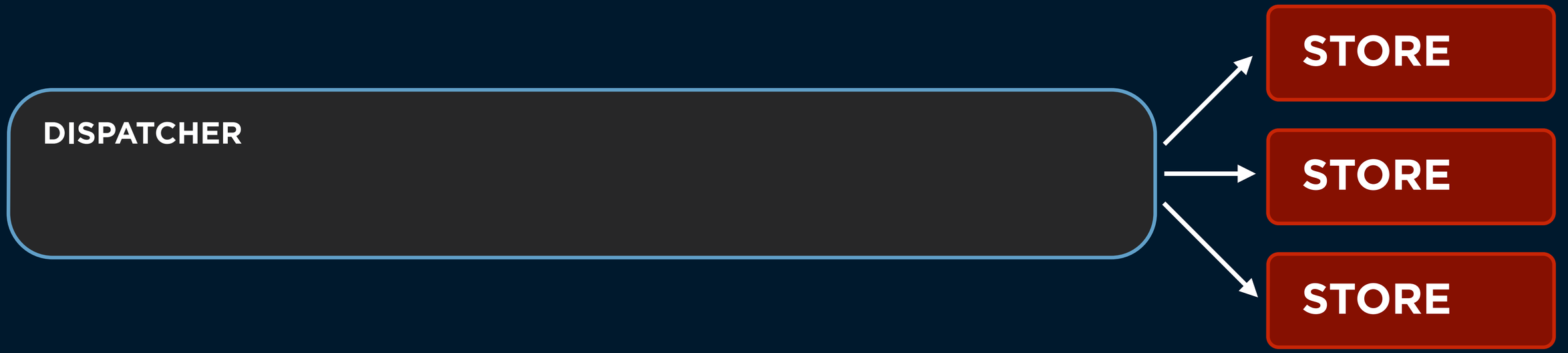


STORE

STORE

STORE

Already partway there!



Writes:

~~mutate state in place~~

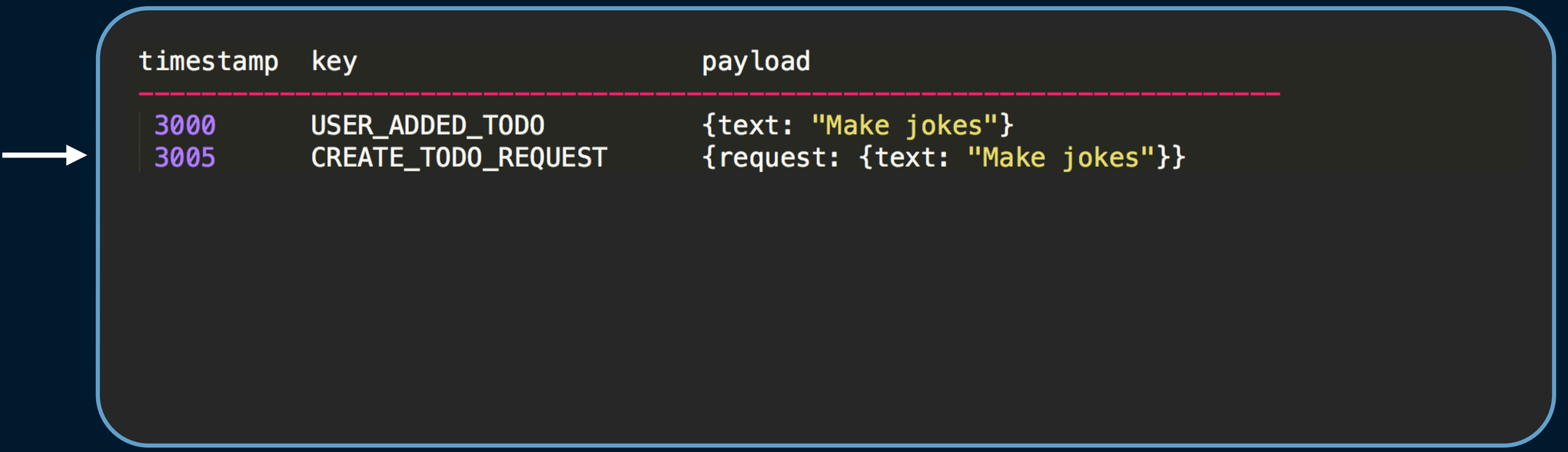
Writes:

append-only stream of immutable facts

Record facts

timestamp	key	payload
3000	USER_ADDED_TODO	{text: "Make jokes"}

Record facts



Record facts

timestamp	key	payload
3000	USER_ADDED_TODO	{text: "Make jokes"}
3005	CREATE_TODO_REQUEST	{request: {text: "Make jokes"}}
3500	CREATE_TODO_RESPONSE	{request: {text: "Make jokes"}}, response: {id: 14}}
6000	USER_ADDED_TODO	{text: "Explain ideas clearly"}
8000	USER_ADDED_TODO	{text: "Cat memes"}
...		
12000	TODOS_LIST_RESPONSE	[{id: 14, text: "Make funny jokes"}]

Record facts

timestamp	key	payload
3000	USER_ADDED_TODO	{text: "Make jokes"}
3005	CREATE_TODO_REQUEST	{request: {text: "Make jokes"}}
3500	CREATE_TODO_RESPONSE	{request: {text: "Make jokes"}}, response: {id: 14}}
6000	USER_ADDED_TODO	{text: "Explain ideas clearly"}
8000	USER_ADDED_TODO	{text: "Cat memes"}
...		
12000	TODOS_LIST_RESPONSE	[{id: 14, text: "Make funny jokes"}]

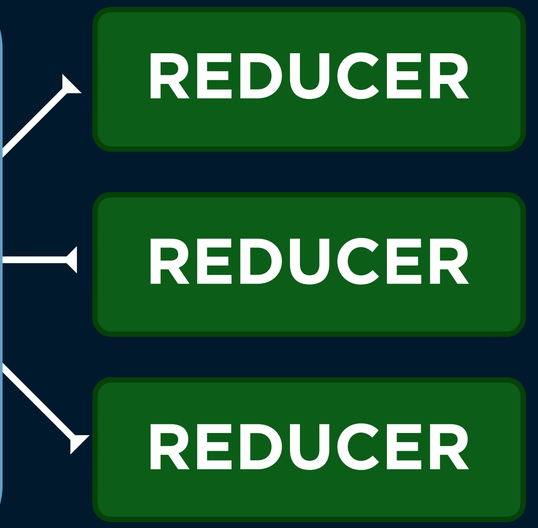
STORE

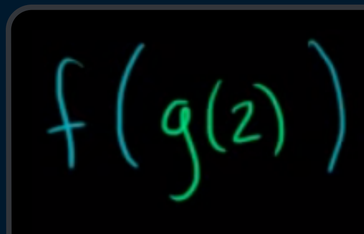
STORE

STORE

Compute views

```
timestamp   key                       payload
-------------------------------------------------------------------------------
 3000       USER_ADDED_TODO           {text: "Make jokes"}
 3005       CREATE_TODO_REQUEST       {request: {text: "Make jokes"}}
 3500       CREATE_TODO_RESPONSE      {request: {text: "Make jokes"}}, response: {id: 14}}
 6000       USER_ADDED_TODO           {text: "Explain ideas clearly"}
 8000       USER_ADDED_TODO           {text: "Cat memes"}
 ...
12000       TODOS_LIST_RESPONSE       [{id: 14, text: "Make funny jokes"}]
```





Reducers

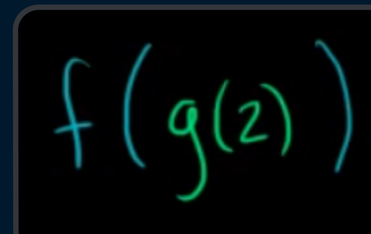
Stores are a function of the actions fired on them

```
f(state, [...actions]) → newState
```

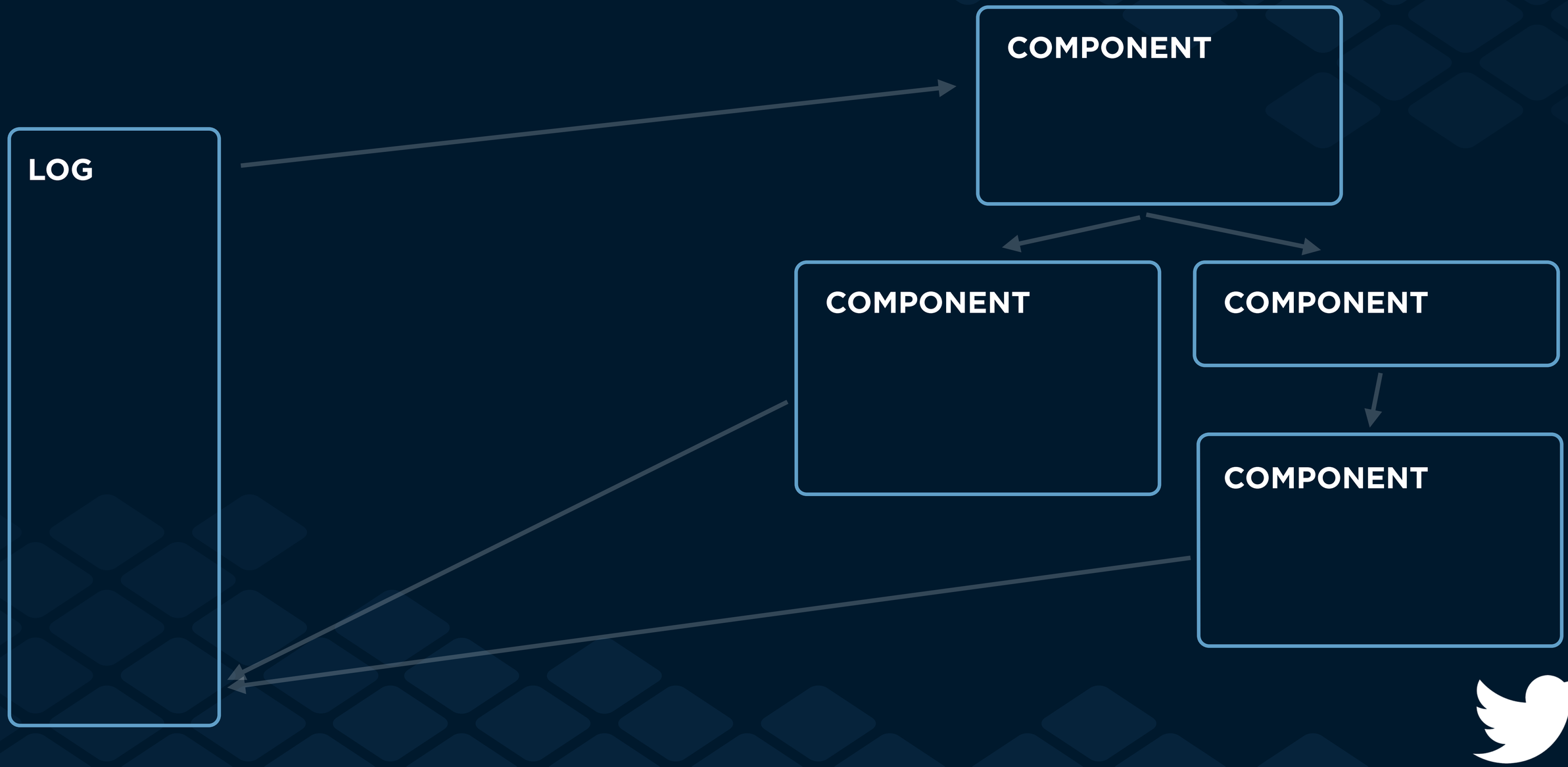
https://speakerdeck.com/jmorrell/jsconf-uy-flux-those-who-forget-the-past-dot-dot-dot-1

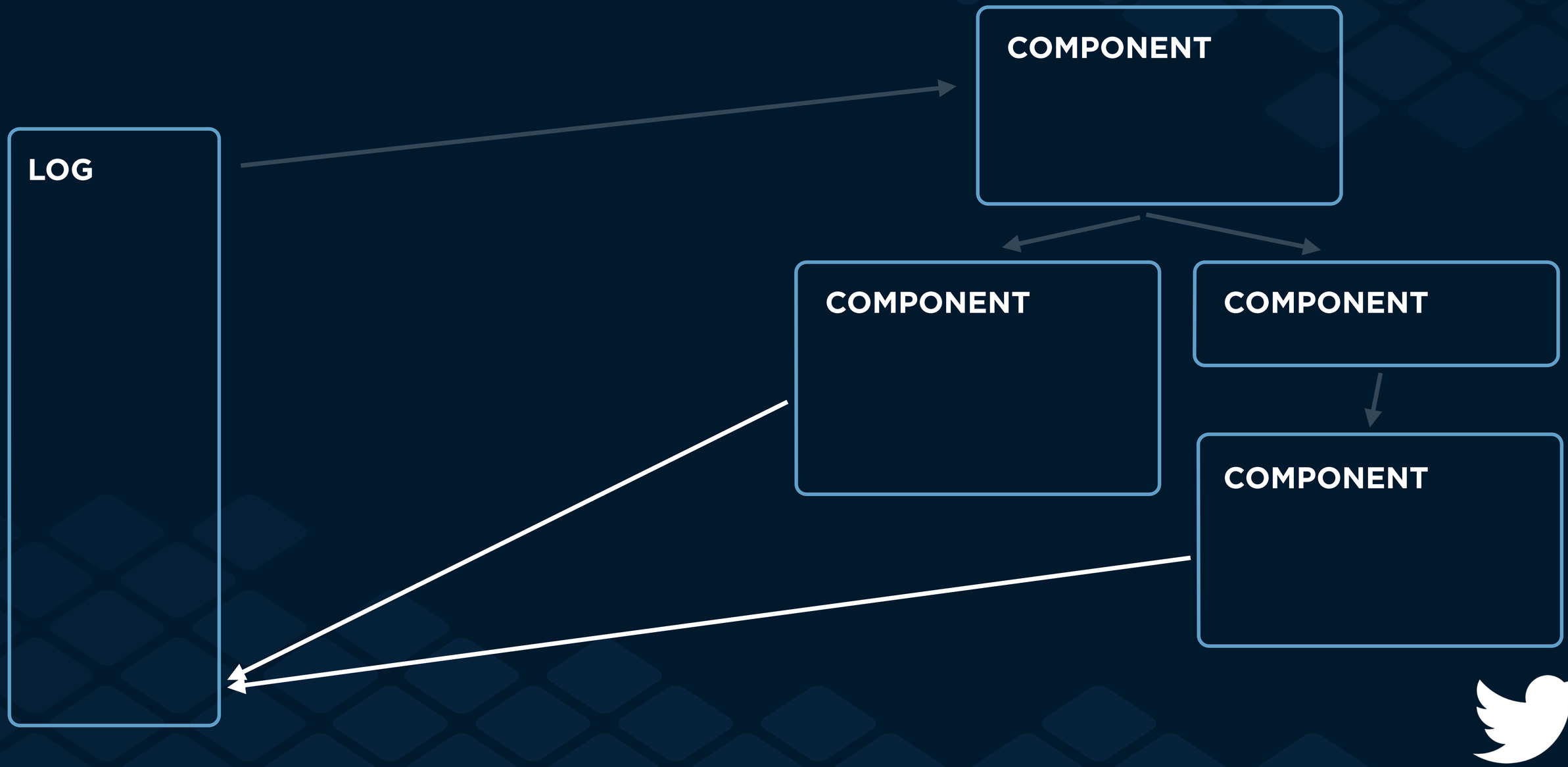
https://github.com/gaearon/redux

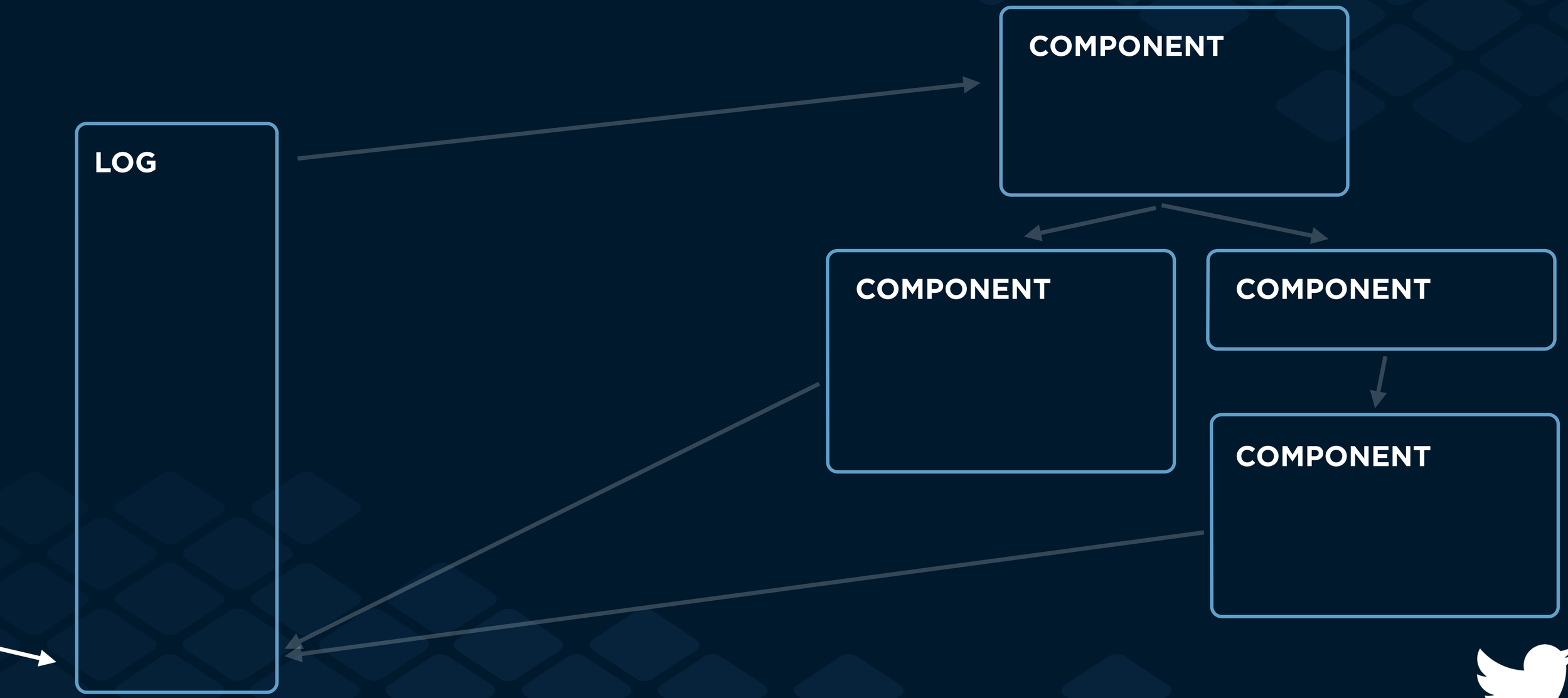
https://github.com/threepointone/disto

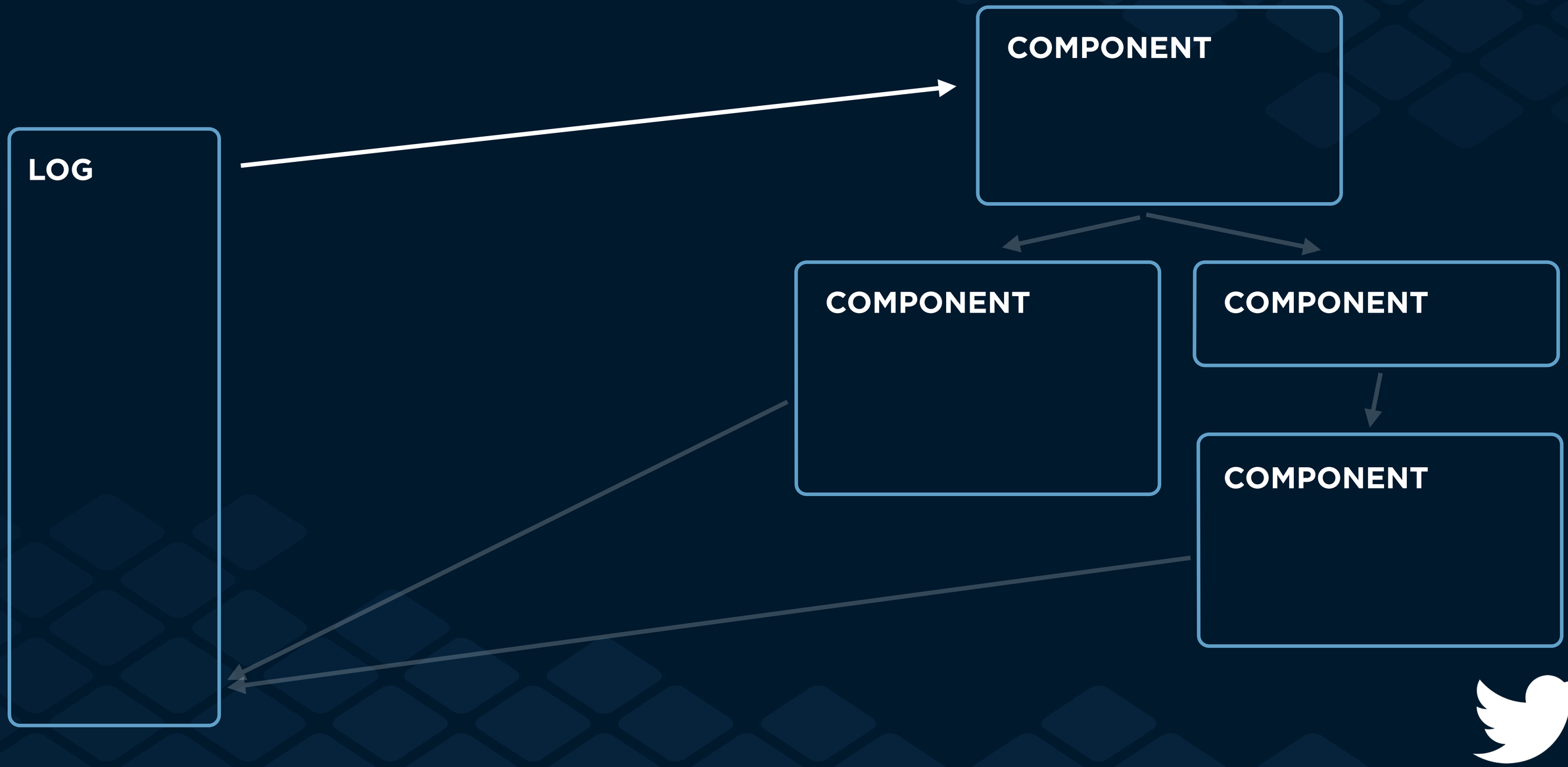


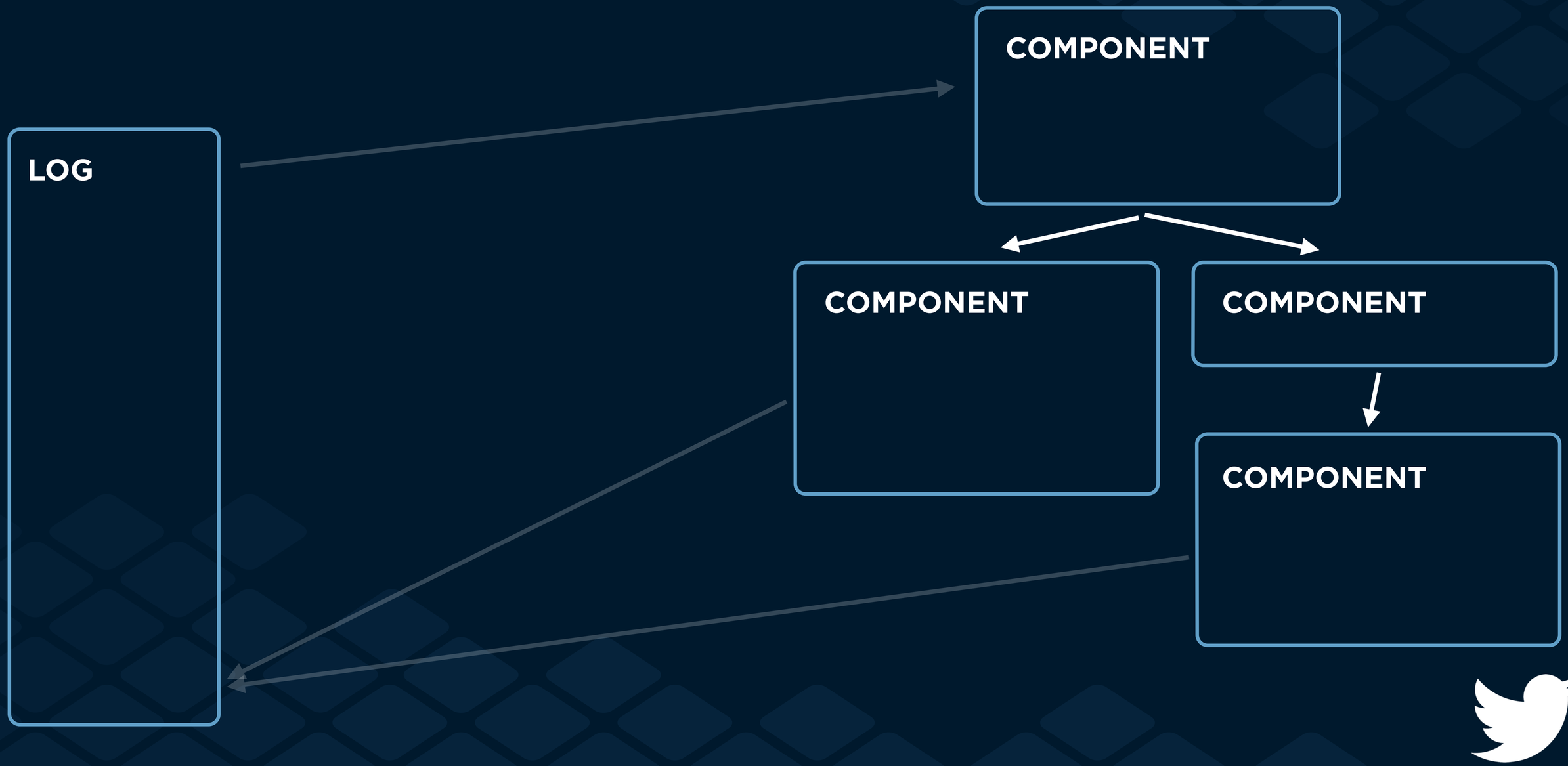
To the whiteboard!

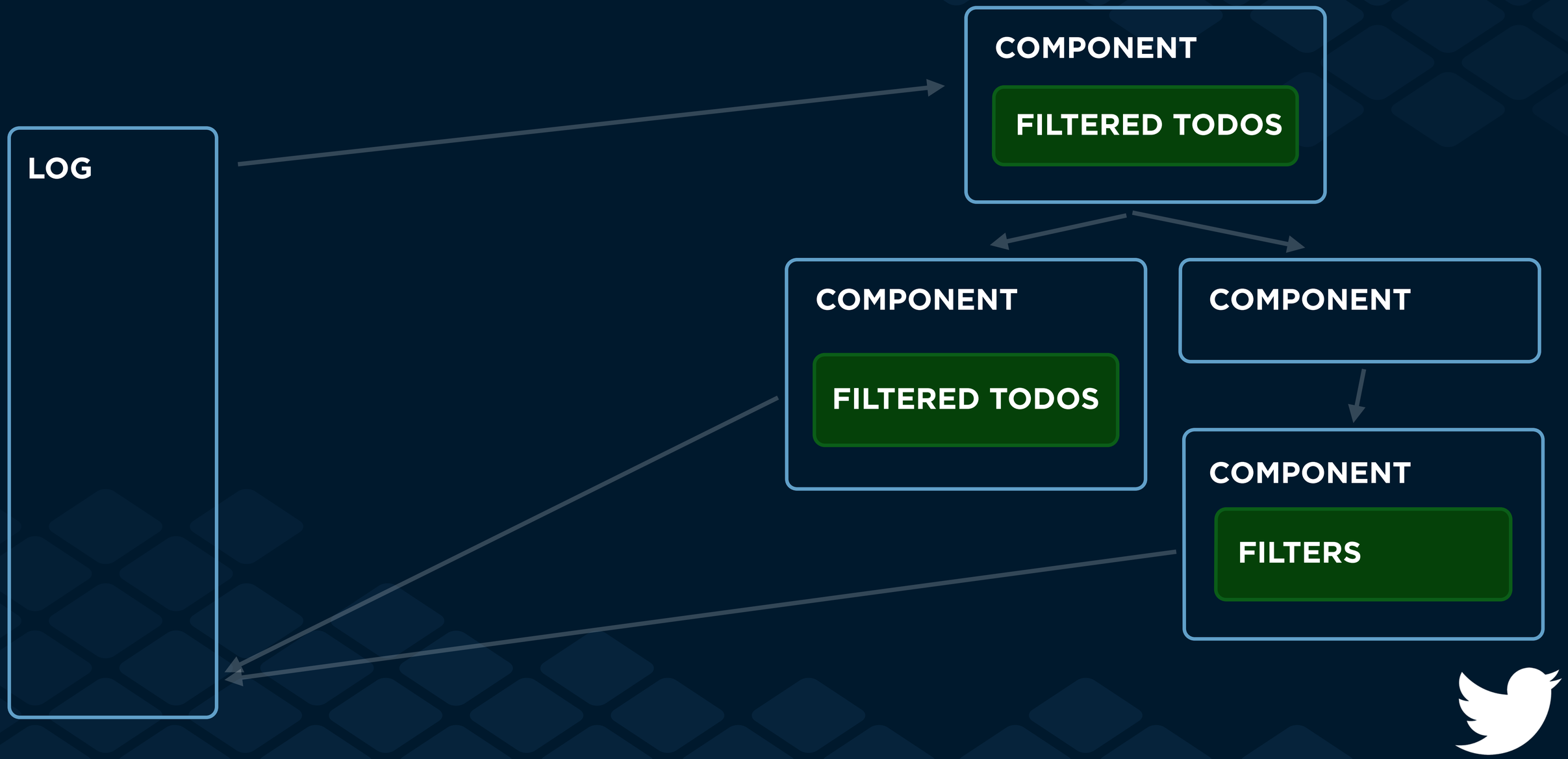

COMPONENT
LOG
COMPONENT
COMPONENT
COMPONENT

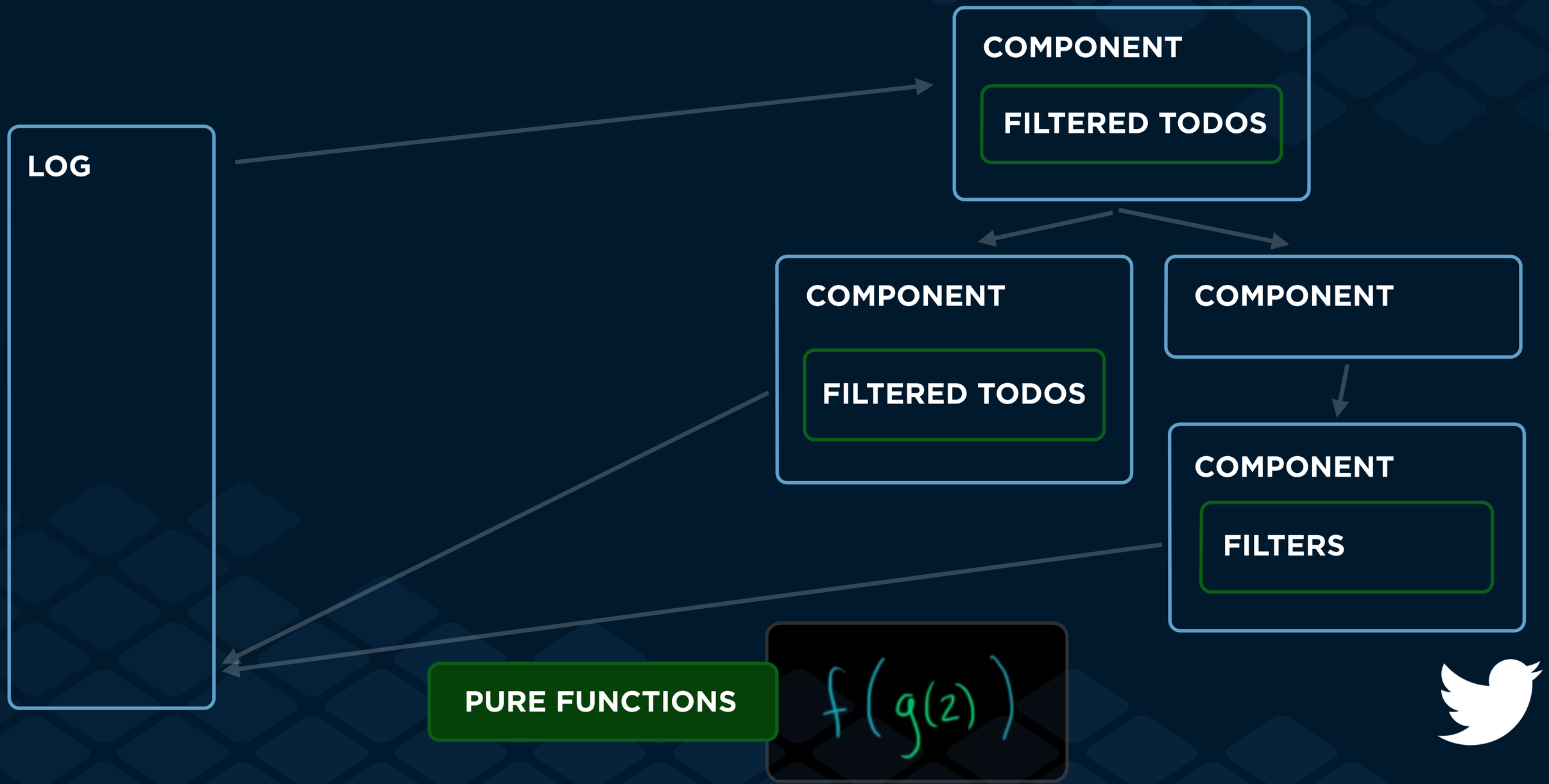

COMPONENT
LOG
COMPONENT
COMPONENT
COMPONENT


COMPONENT
LOG
COMPONENT
COMPONENT
COMPONENT


COMPONENT
LOG
COMPONENT
COMPONENT
COMPONENT


COMPONENT
LOG
COMPONENT
COMPONENT
COMPONENT


LOG
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS


LOG
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS
f(g(2))

Challenges

Challenges

Building blocks

Optimizations

Building blocks

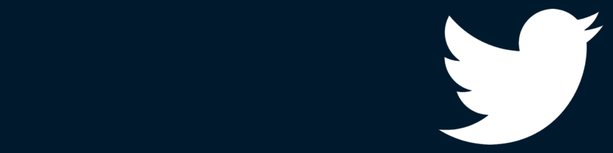
LOG

FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT

HTTP REQUESTS

USER ACTIONS

REALTIME UPDATES



Building blocks

LOG

FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT

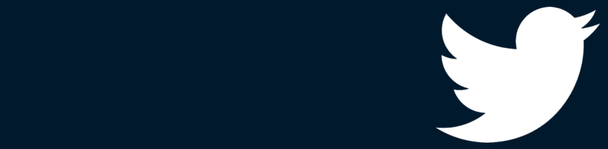
HTTP REQUESTS

USER ACTIONS

REALTIME UPDATES

TODOS

FILTERS

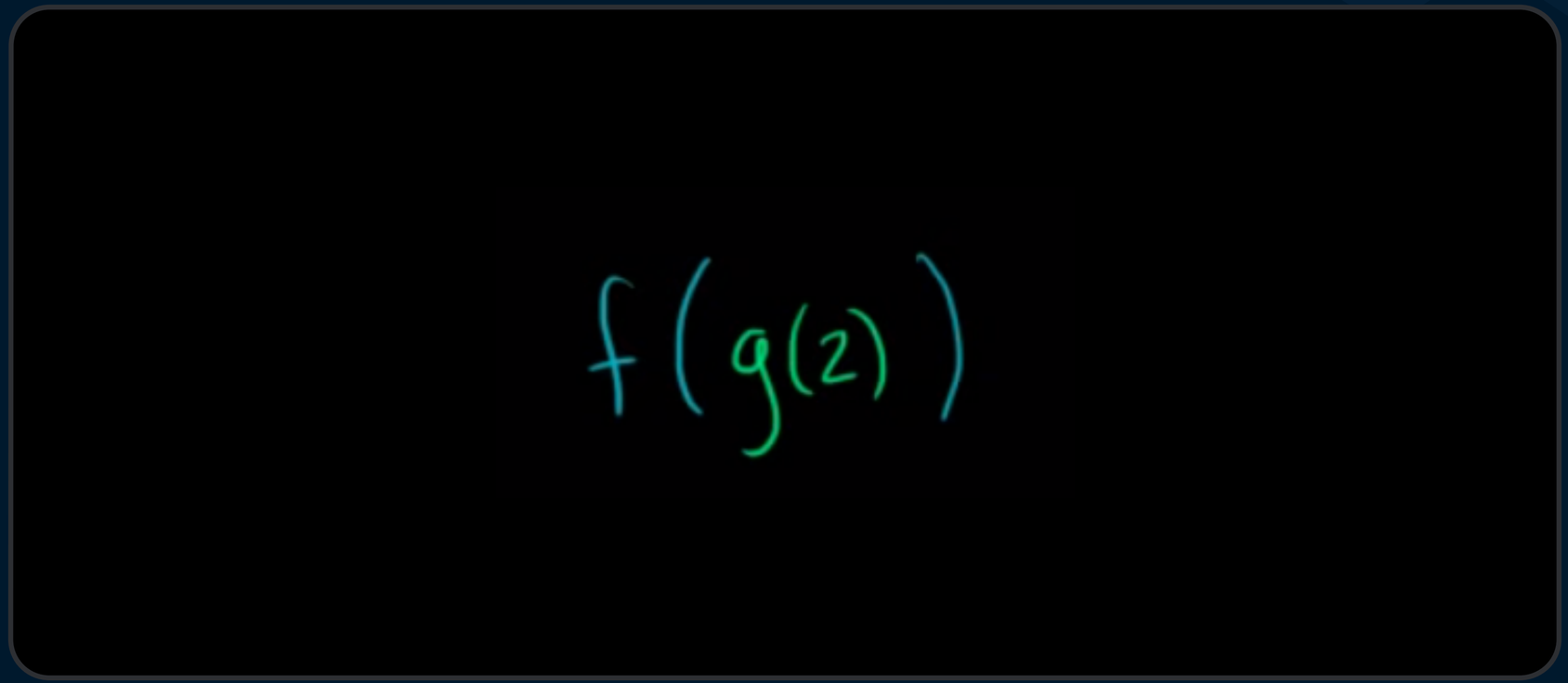


Building blocks

Building blocks

LOG

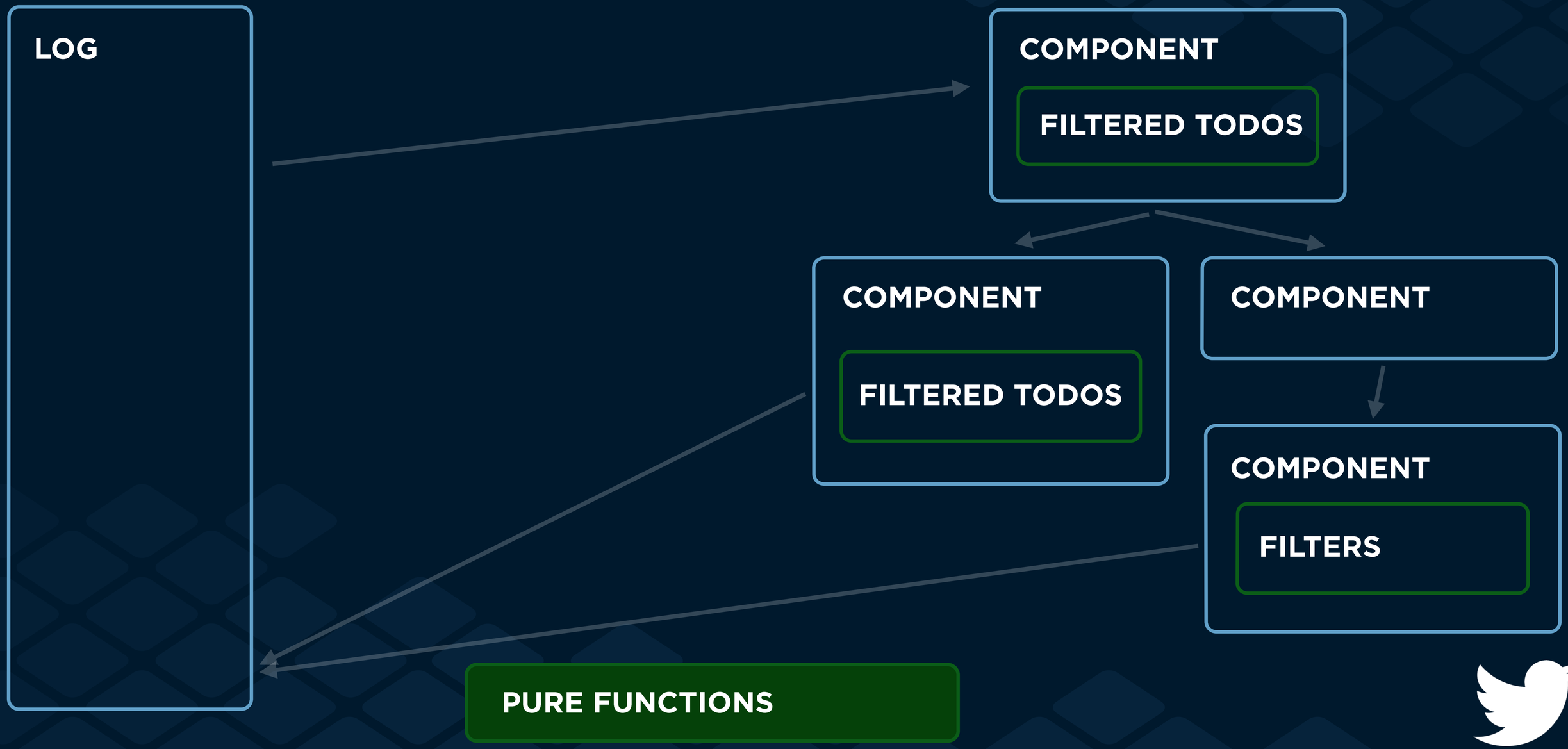
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT

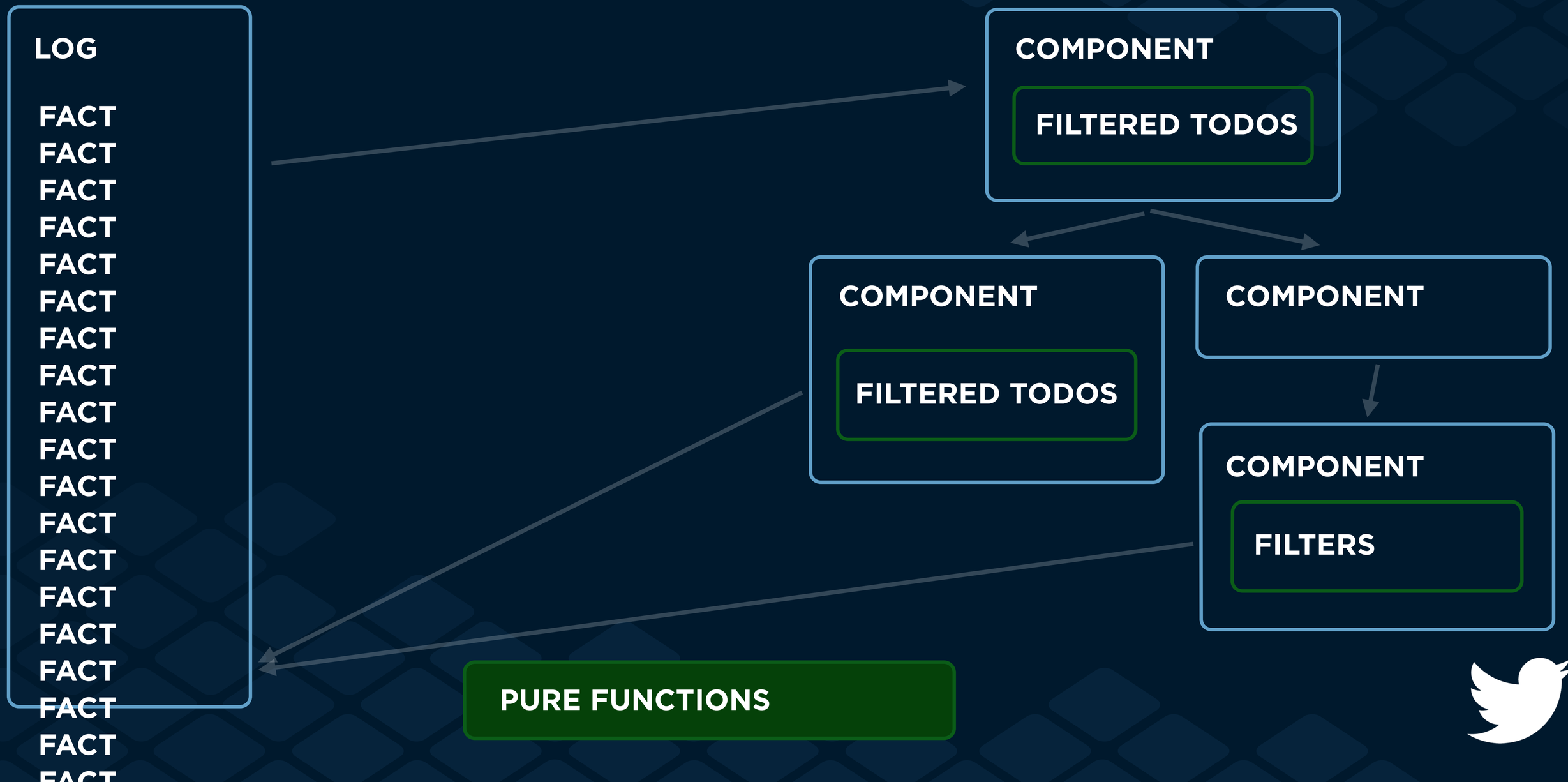


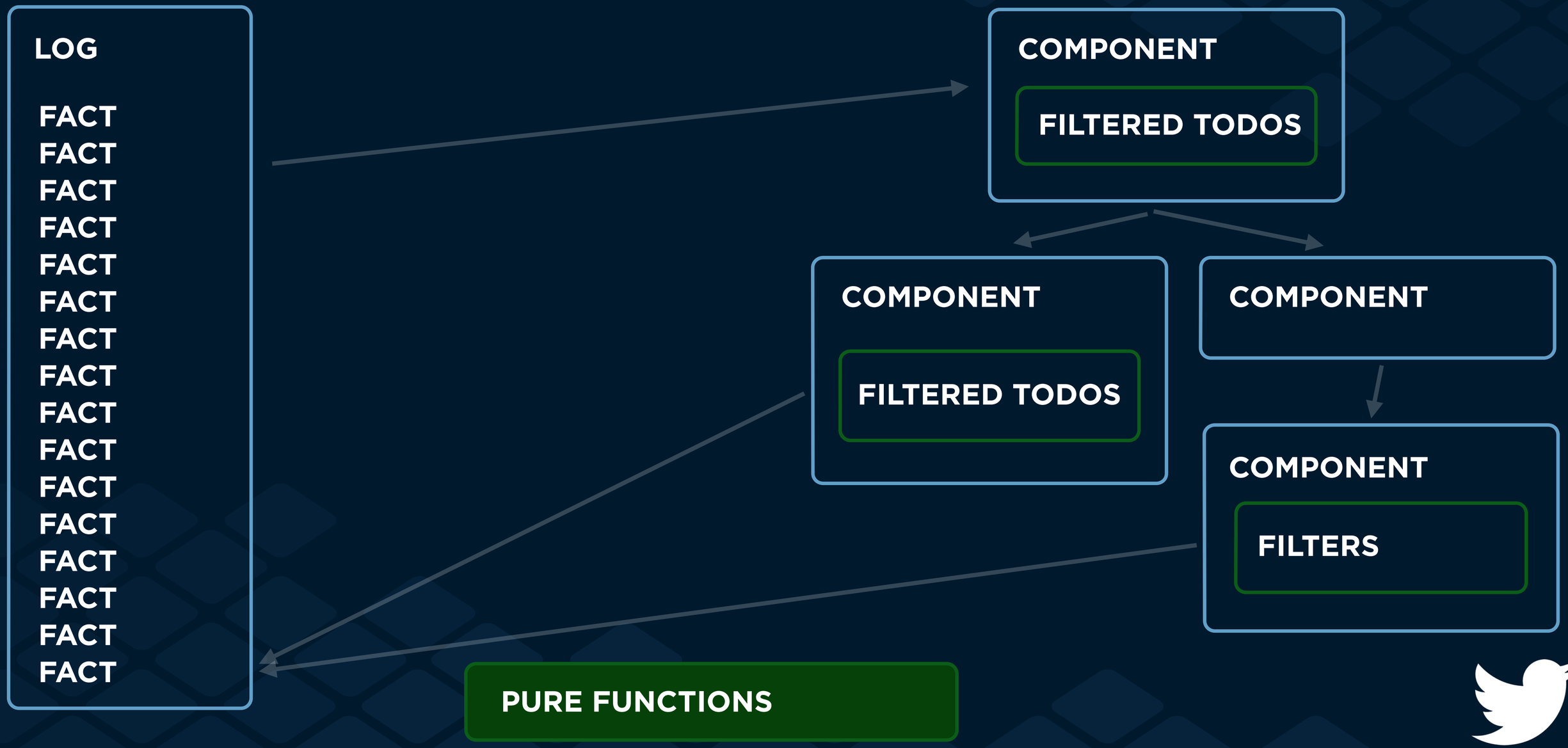
Challenges

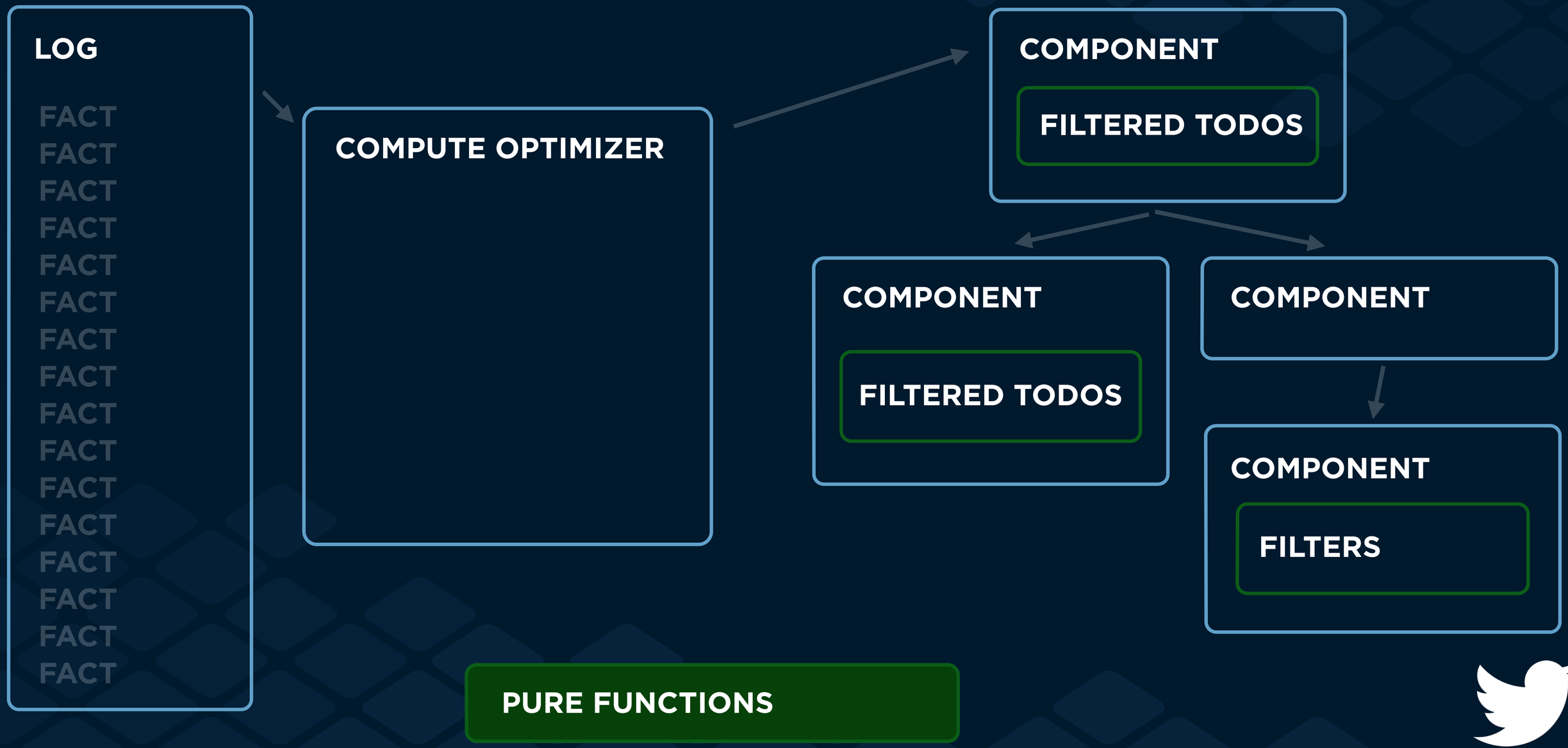
Building blocks

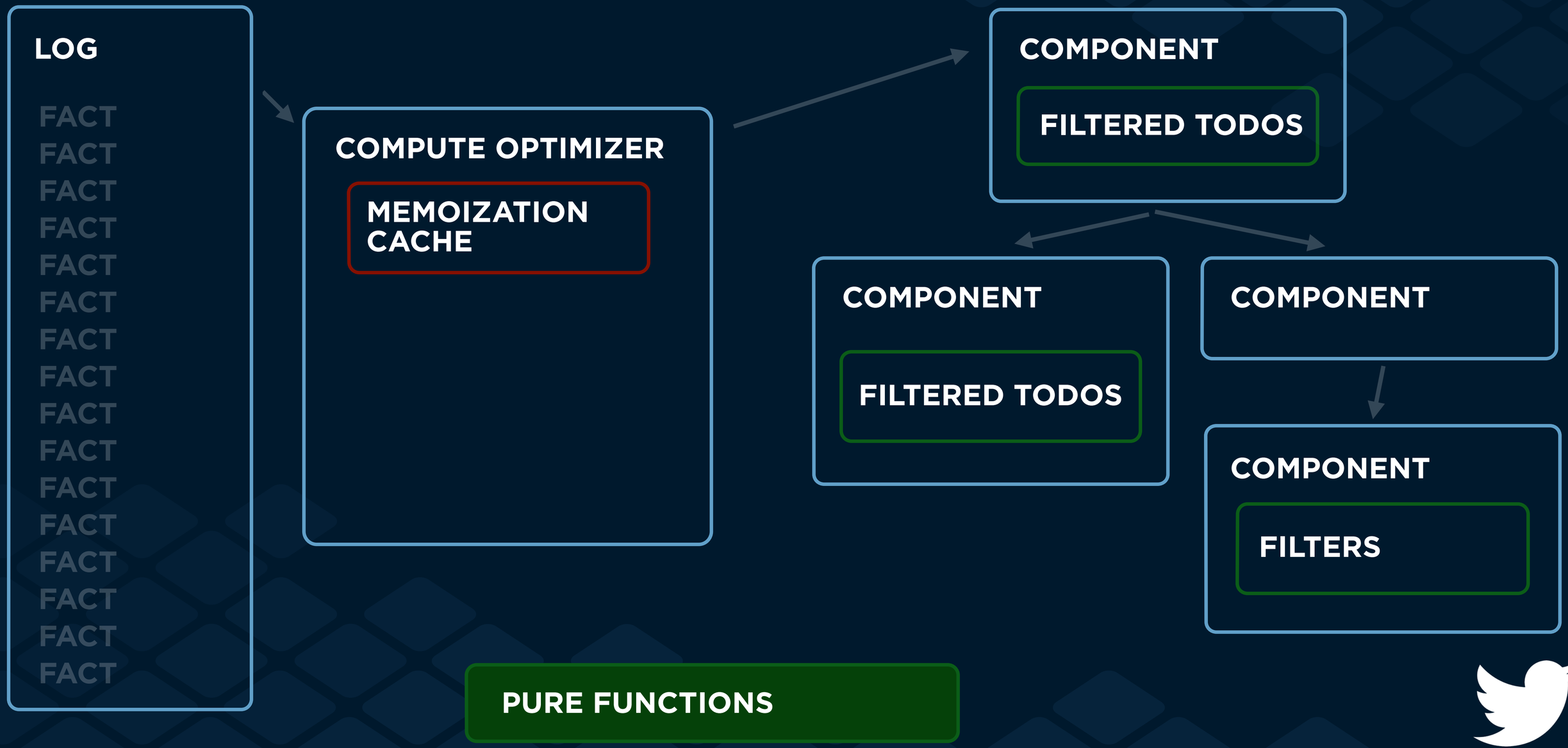
Optimizations

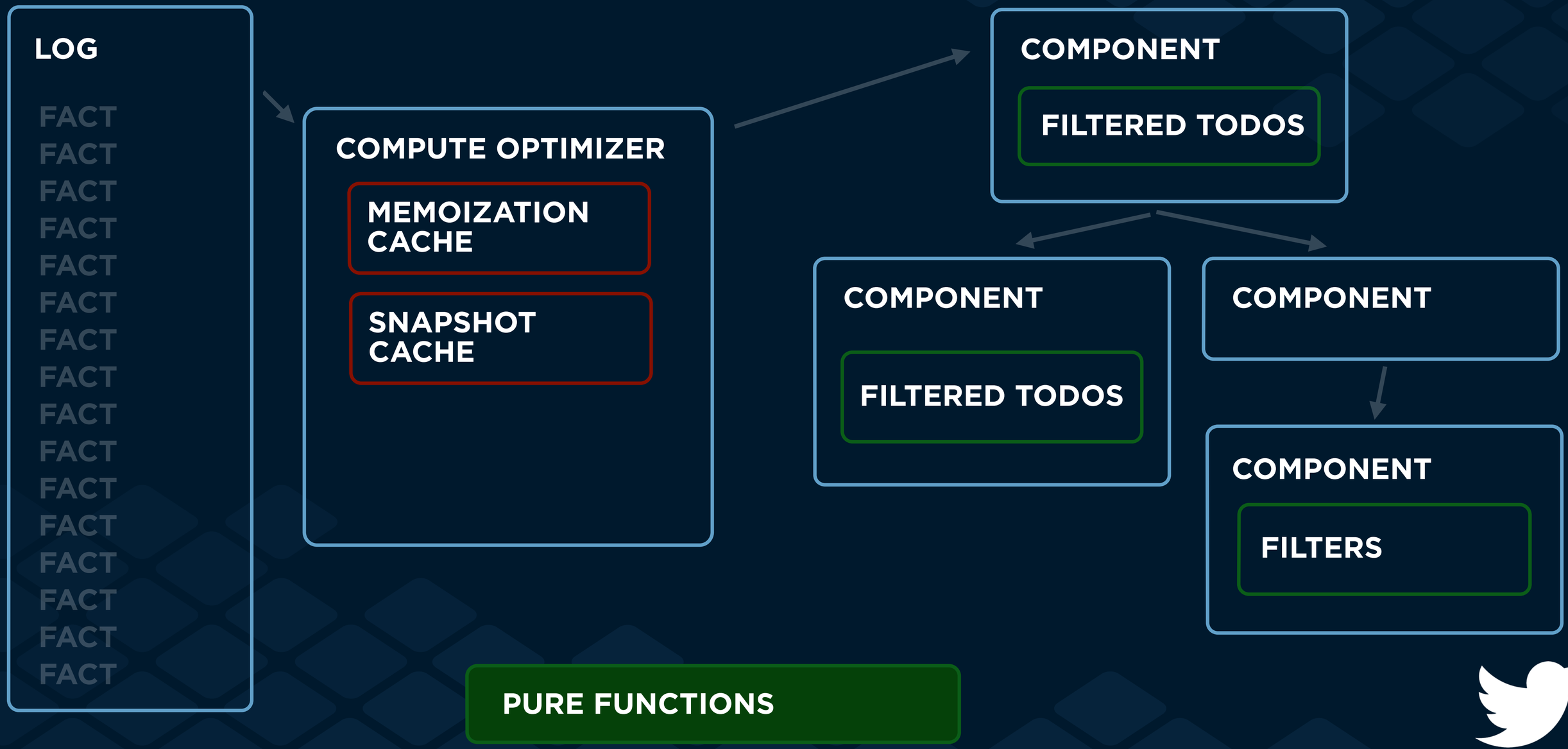

LOG
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS

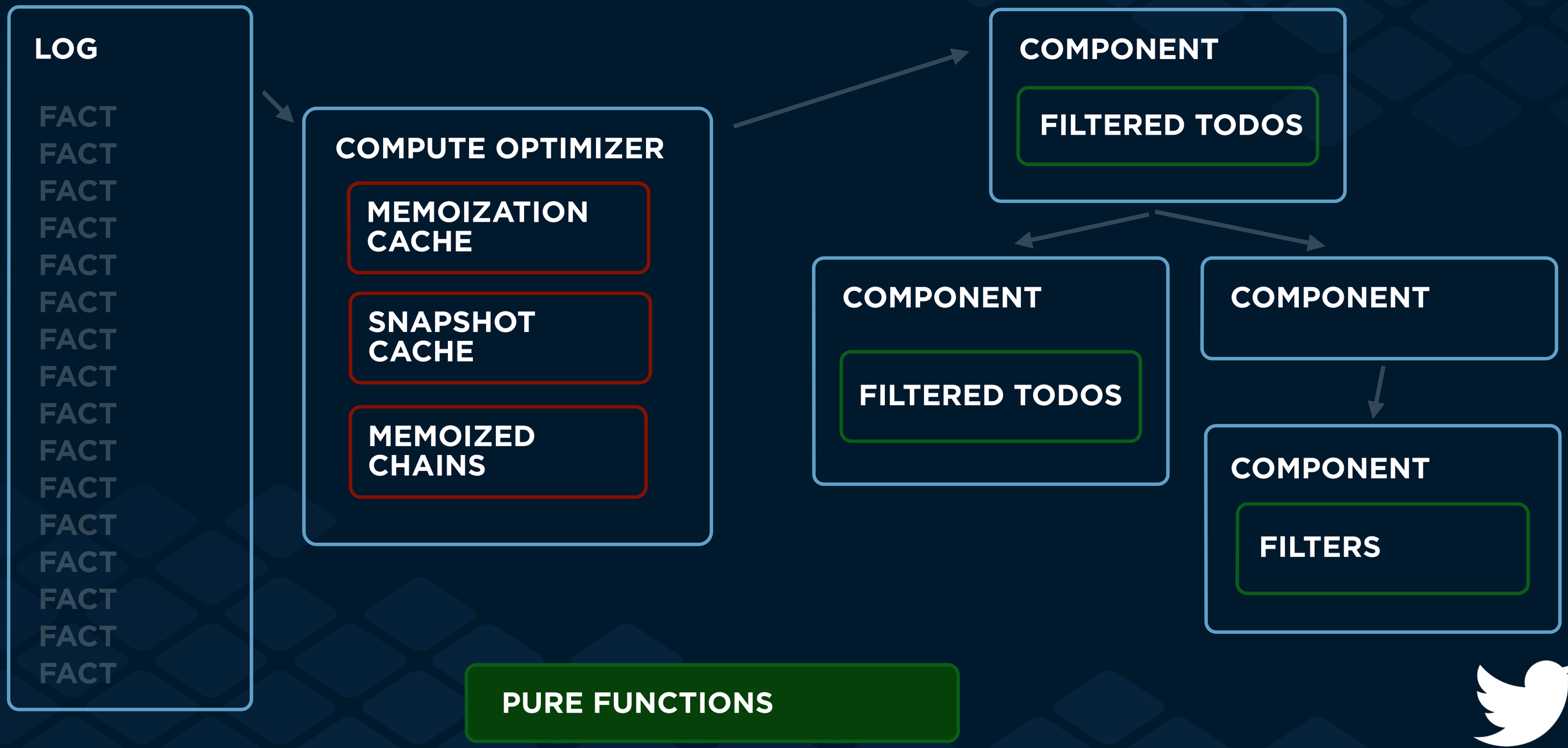

LOG
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS

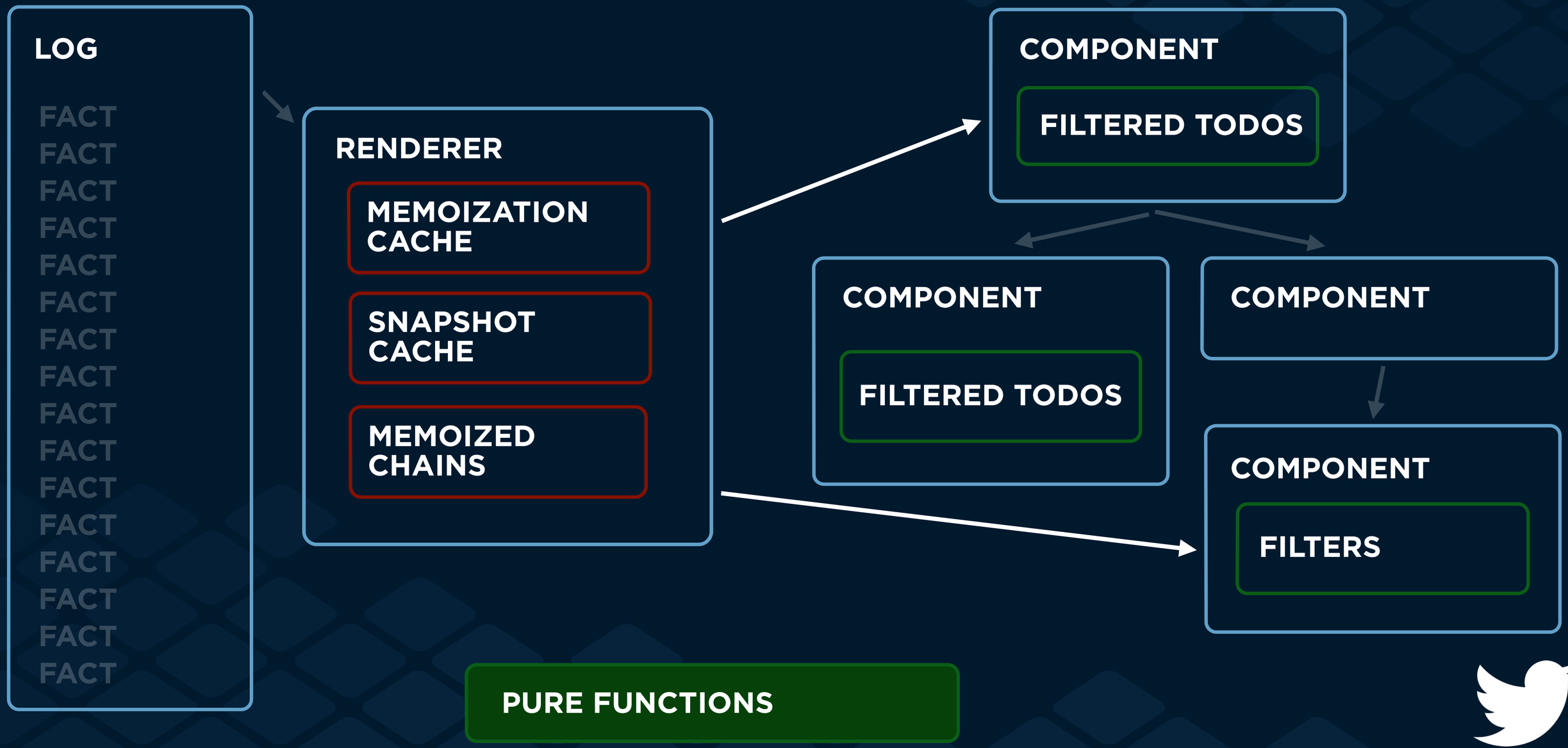

LOG
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS


LOG
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
COMPUTE OPTIMIZER
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS


LOG
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
COMPUTE OPTIMIZER
MEMOIZATION CACHE
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS


LOG
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
COMPUTE OPTIMIZER
MEMOIZATION CACHE
SNAPSHOT CACHE
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS


LOG
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
COMPUTE OPTIMIZER
MEMOIZATION CACHE
SNAPSHOT CACHE
MEMOIZED CHAINS
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS


LOG
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
FACT
RENDERER
MEMOIZATION CACHE
SNAPSHOT CACHE
MEMOIZED CHAINS
COMPONENT
FILTERED TODOS
COMPONENT
FILTERED TODOS
COMPONENT
COMPONENT
FILTERS
PURE FUNCTIONS

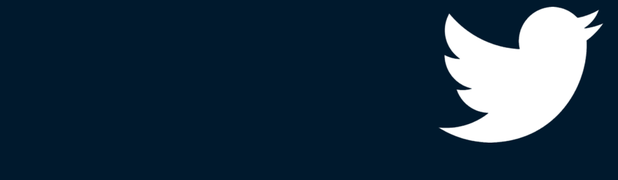
Takeaways

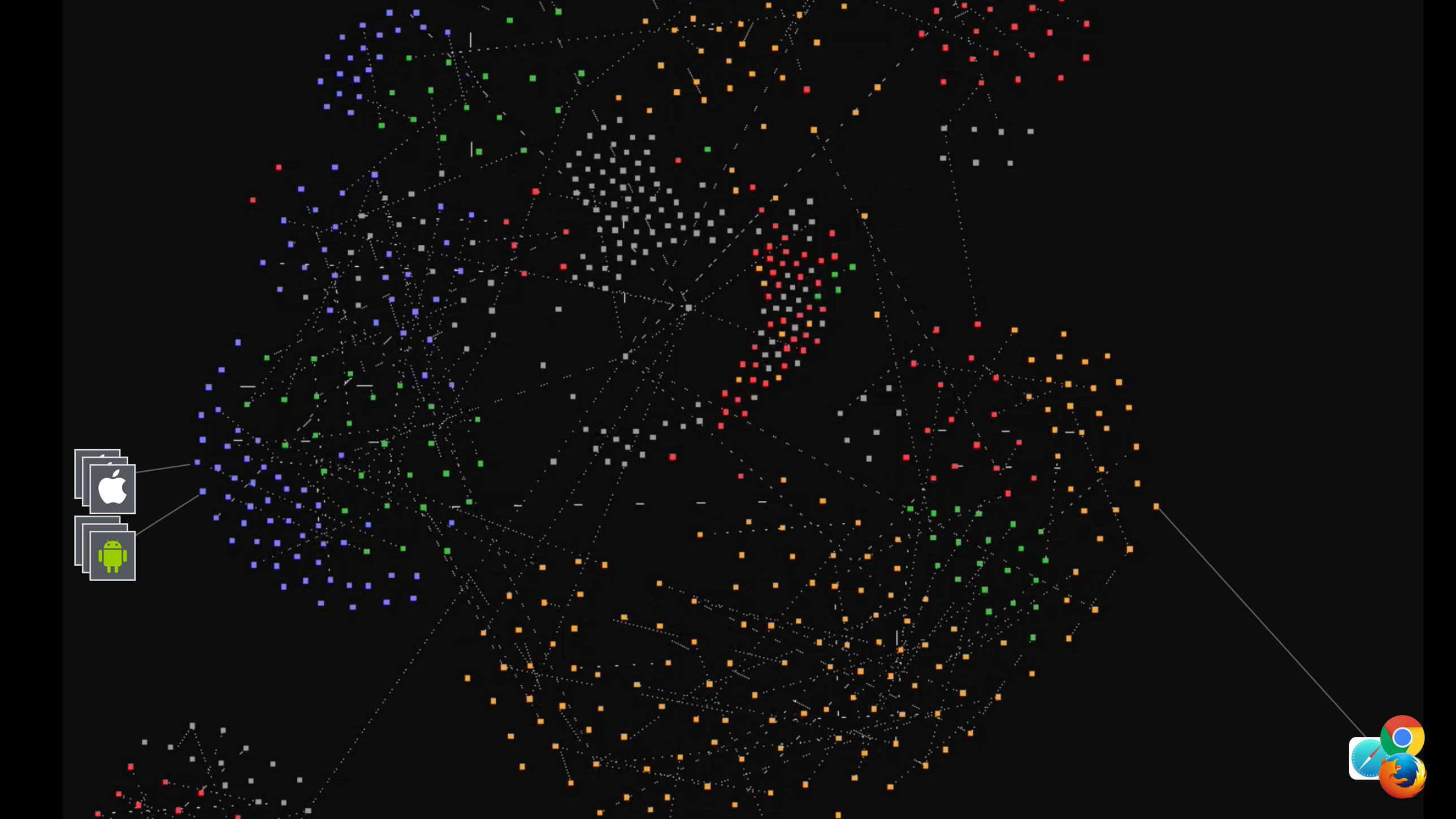
Simplifying

Collaborating across the stack

Push effects out to the edge

Embrace the log





Joseph Savona
@en_JS

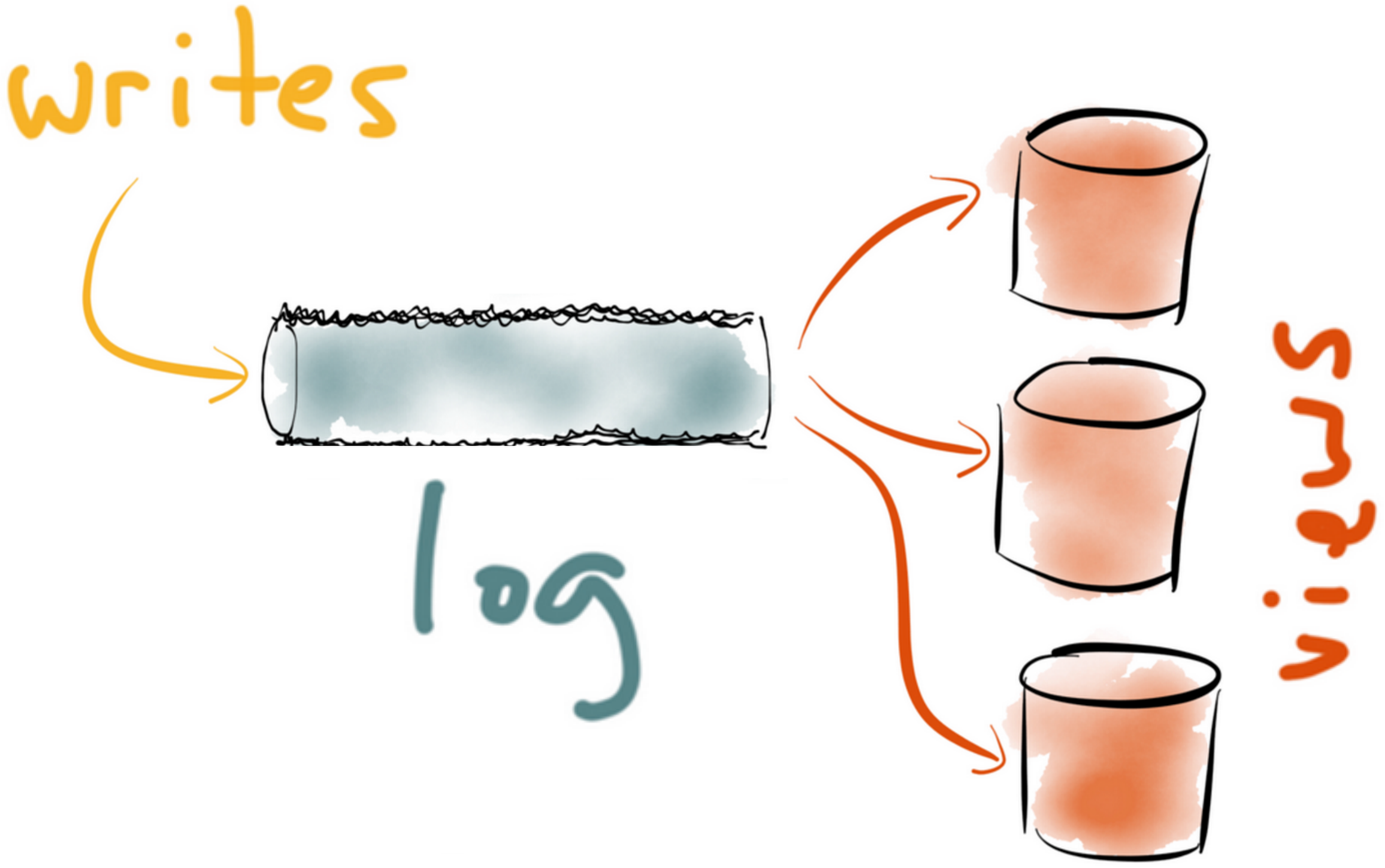
Following

The solution to async UI development is not async APIs: declarative, functional approaches are more powerful and easier to reason about.

RETWEETS 9

FAVORITES 16

4:57 PM - 11 Jun 2015


writes
log
views

Thanks!

@krob