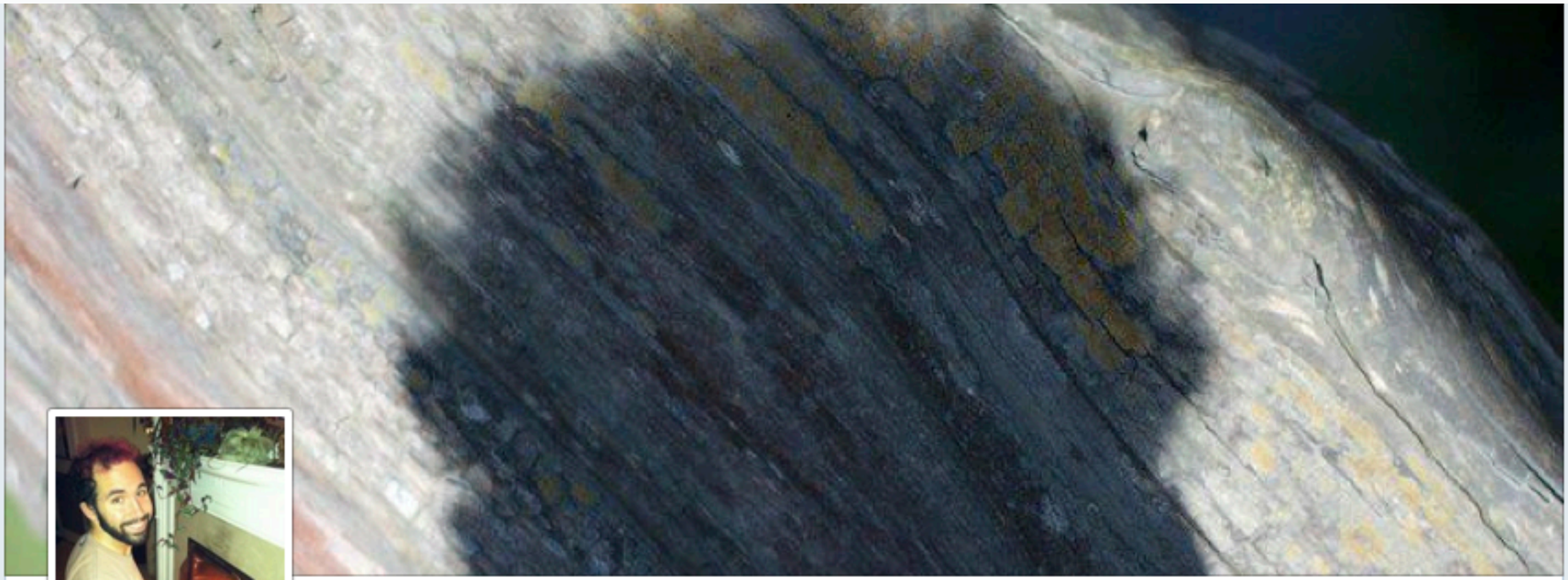
Stanford ACM
January 20, 2012

Jack Stahl
Kris Rasmussen

HI.


Jack Stahl
Update Info
Activity Log
♥ at Asana
Studied Computer Science at Stanford University
Lives in San Francisco, California
In a relationship with Erin Michelle Chalfant
9
About
Friends 827
Photos 391
Map 71
Likes 191

I'M JACK.

Wall
Info
Photos (440)
Friends
Subscriptions (1)

In a relationship with

Kris Rasmussen ○

✔ Frie

Engineer at Asana · Studied Computer Science at UCLA · Lives in Atherton, California · In a relationship with Becky Christensen · From Atherton, California
Born on November 22, 1982

Work and Education

Employers

Asana
Engineer · Apr 2010 to present

Microsoft
Software engineer · Redmond, Washington

Aptana
Chief Architect · San Mateo, California

HE'S KRIS.

WE WORK @

asana:

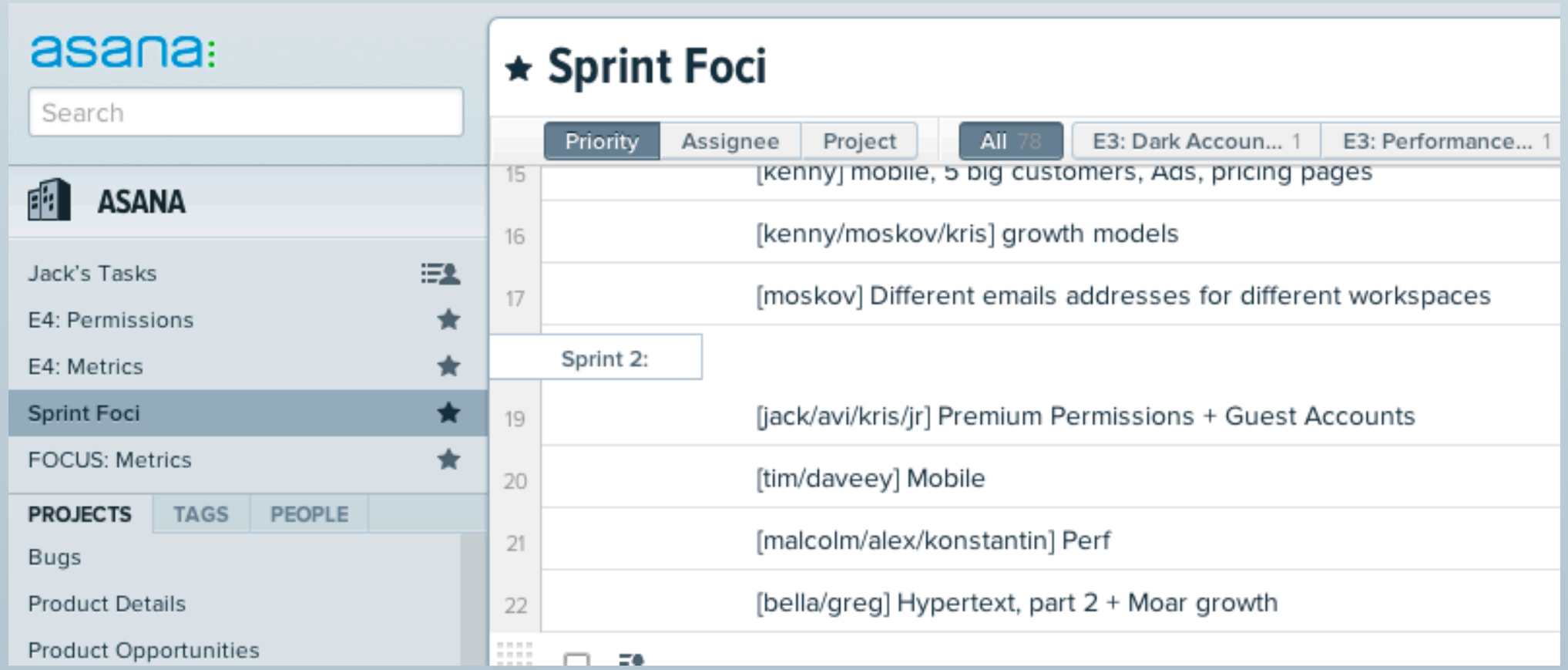
WHY DO WE DO WHAT WE DO?

IMPACT.

IMPACT.

Product

PRODUCT


asana:
Search
ASANA
Jack's Tasks
E4: Permissions
E4: Metrics
Sprint Foci
FOCUS: Metrics
PROJECTS
TAGS
PEOPLE
Bugs
Product Details
Product Opportunities
★ Sprint Foci
Priority
Assignee
Project
All 78
E3: Dark Accoun... 1
E3: Performance... 1
15
[kenny] mobile, 5 big customers, Ads, pricing pages
16
[kenny/moskov/kris] growth models
17
[moskov] Different emails addresses for different workspaces
Sprint 2:
19
[jack/avi/kris/jr] Premium Permissions + Guest Accounts
20
[tim/daveey] Mobile
21
[malcolm/alex/konstantin] Perf
22
[bella/greg] Hypertext, part 2 + Moar growth

PRODUCT

- Leverage: Contribute by helping others make an contribute.

PRODUCT

- Leverage: Contribute by helping others make an contribute.
- Joy: Do it for yourself.

PRODUCT

- Leverage: Contribute by helping others make an contribute.
- Joy: Do it for yourself.
- Integrity: Change the world in accordance with your own values.

IMPACT.

Technology

TECHNOLOGY

- Leverage: Automation and abstraction are awesome.

TECHNOLOGY

- Leverage: Automation and abstraction are awesome.
- Longterm: Encapsulation is essential.

TECHNOLOGY

- Leverage: Automation and abstraction are awesome.
- Longterm: Encapsulation is essential.
- Live it.

</FLUFF>

LUNA

This is how we do it.

REACTIVITY

When data changes, automatically update the parts of the UI that depends on it.

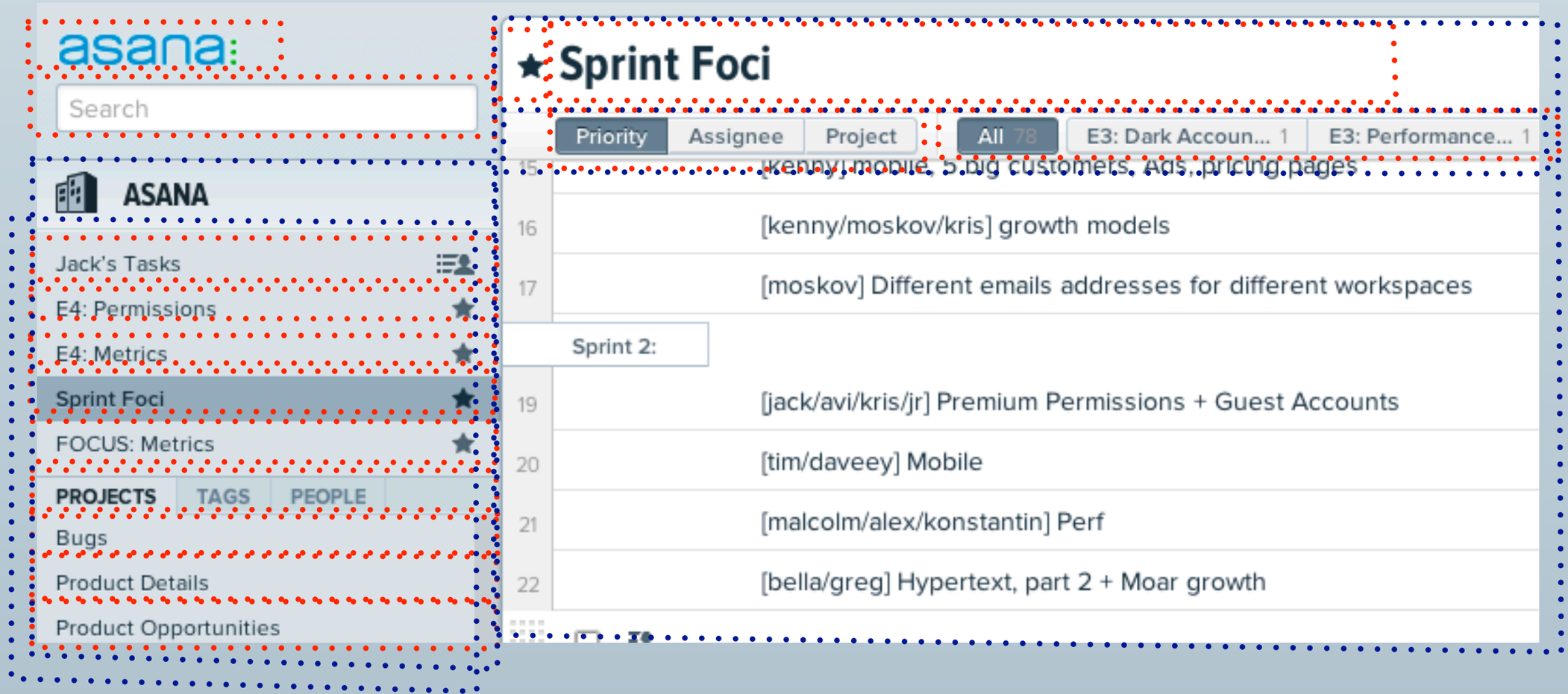
SYNC AND SIMULATION

The server simulates the client, and changes on one are automatically sync’ed to the other.

UNKNOWNS

Formalizing the concept of “waiting on the server to know what to do”.

REACTIVITY



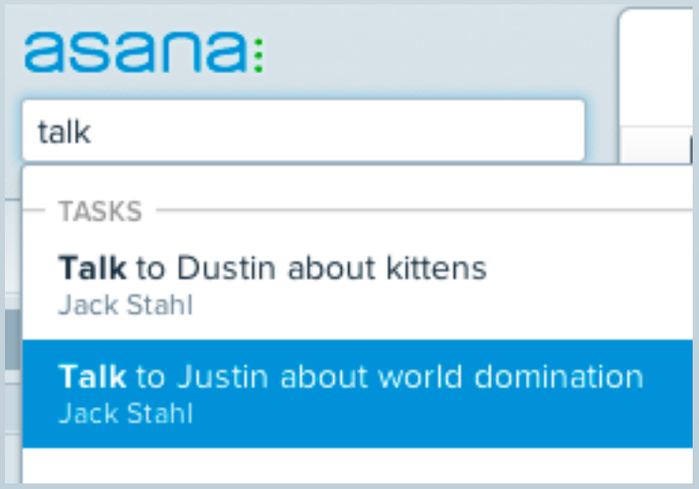
Inner Reactive Boundaries

Outer Reactive Boundaries

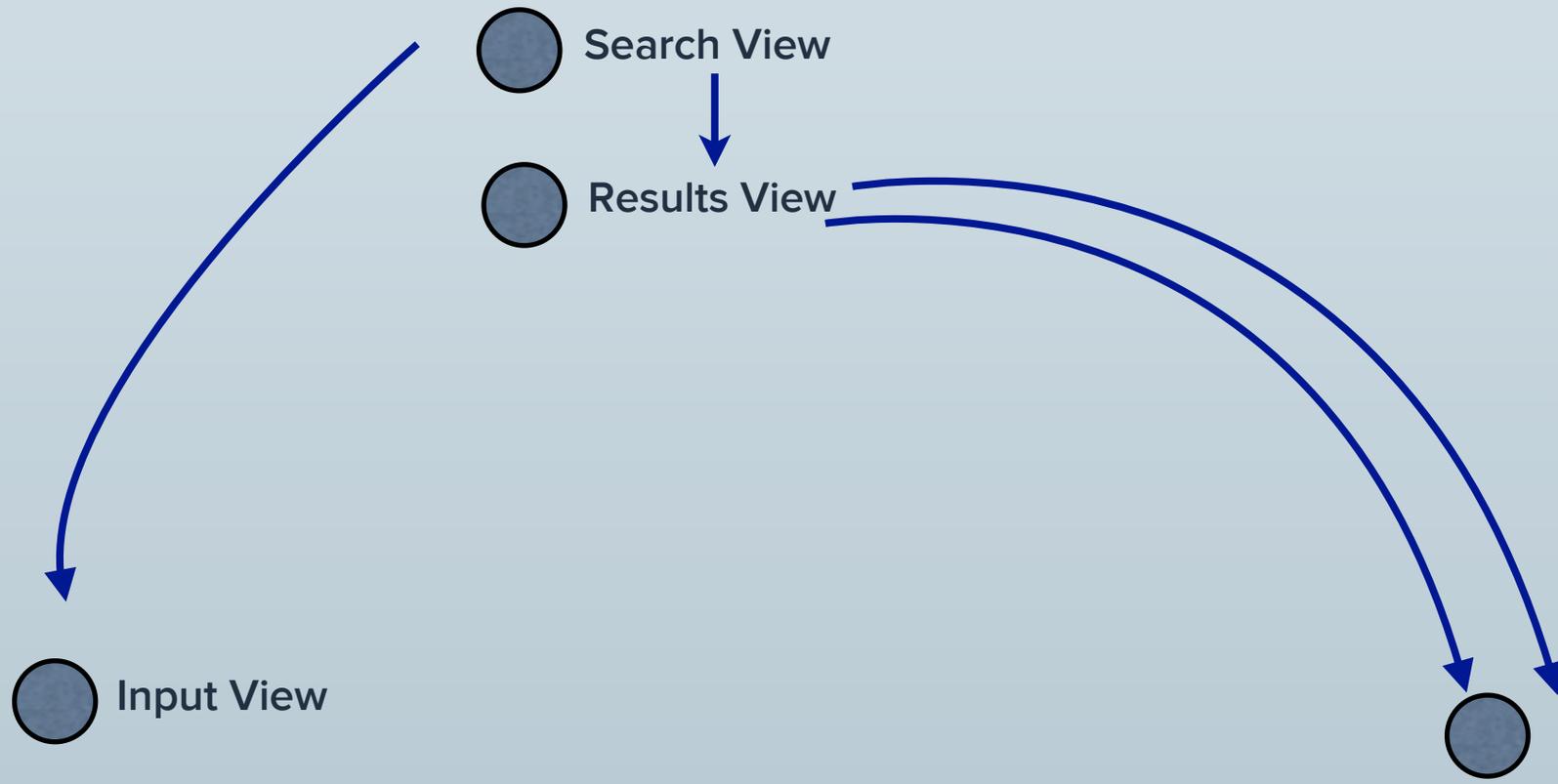
...plus a bunch more, especially in the grid

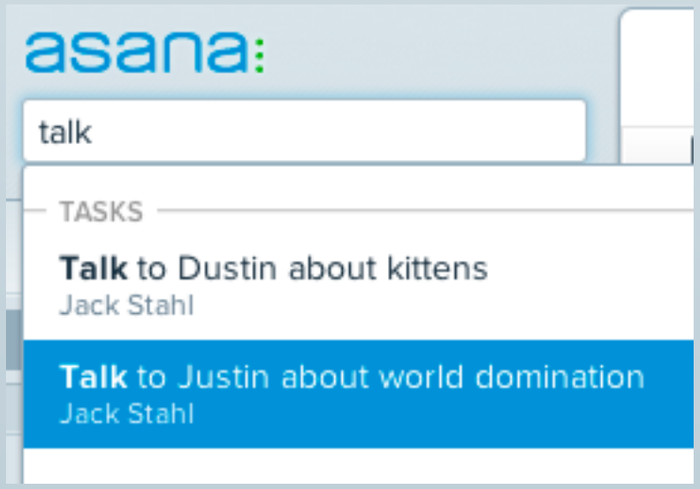
REACTIVITY

Let’s take a look at search.

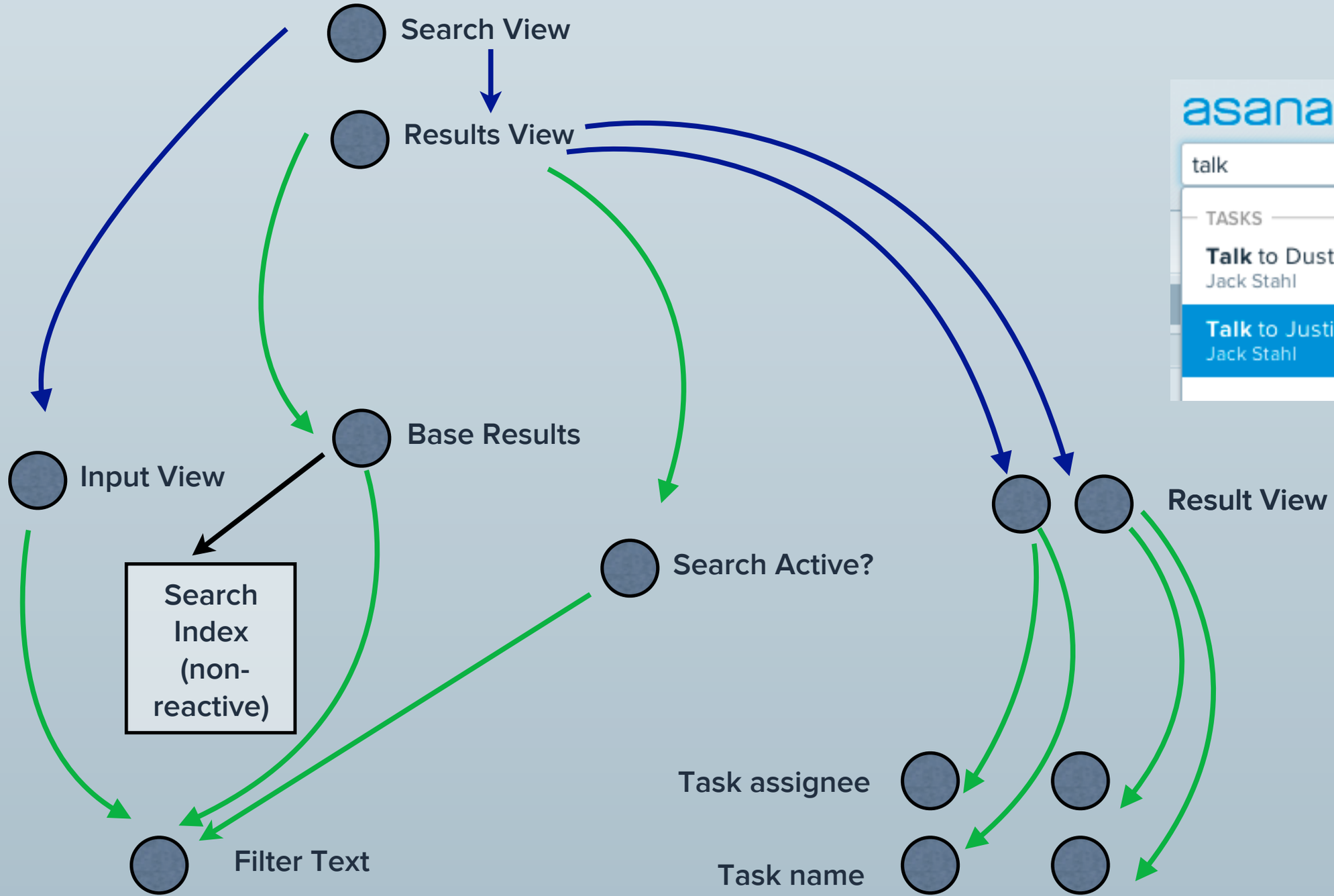


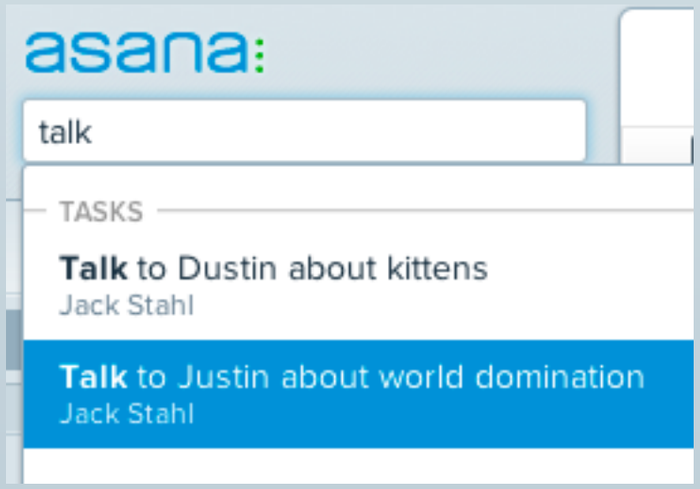
REACTIVITY


Search View
Results View
Input View
Result View

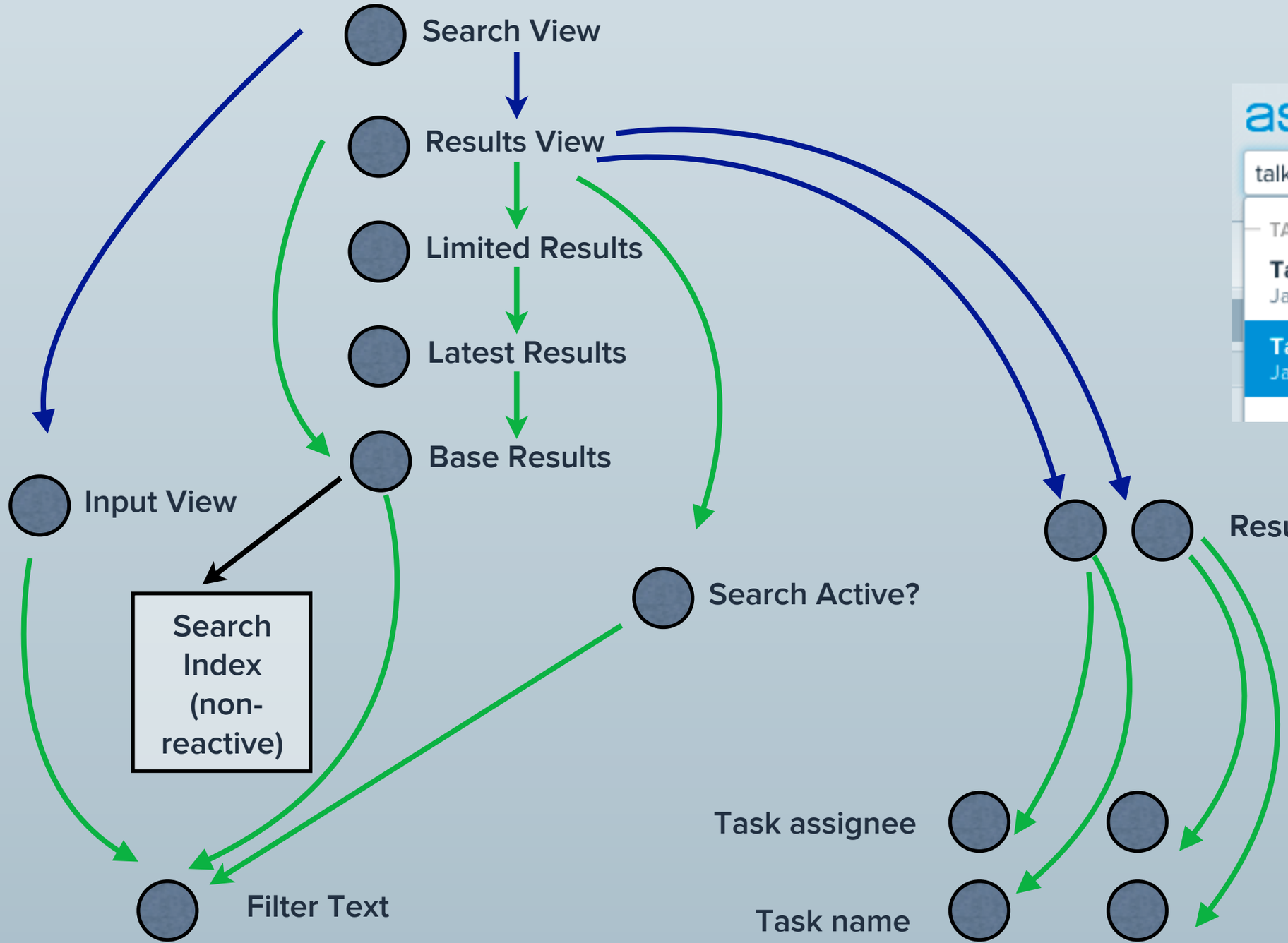

asana:
talk
TASKS
Talk to Dustin about kittens
Jack Stahl
Talk to Justin about world domination
Jack Stahl

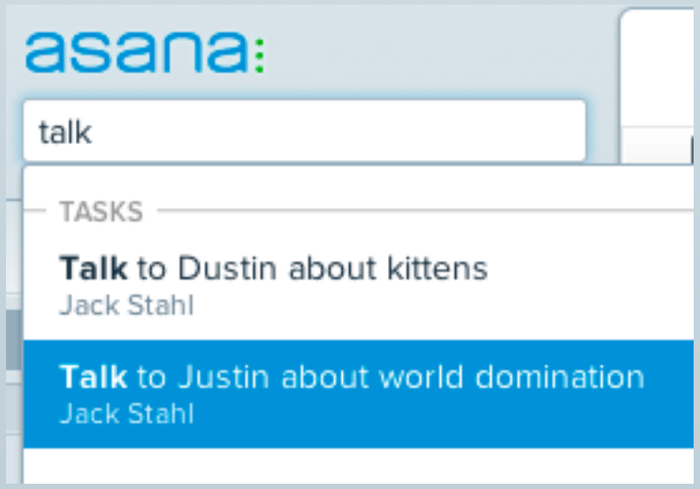
REACTIVITY


Search View
Results View
Base Results
Input View
Search Index (non-reactive)
Search Active?
Filter Text
Task assignee
Task name
Result View


asana:
talk
TASKS
Talk to Dustin about kittens
Jack Stahl
Talk to Justin about world domination
Jack Stahl

REACTIVITY





Reactive Dependency: Child may change in value, causing parent to recompute

Reactive Boundary: No return value, prevent changes in children from causing recomputation of parent

REACTIVITY

TOO FEW BOUNDARIES

TOO MANY BOUNDARIES

REACTIVITY

TOO FEW BOUNDARIES

- Performance: app is less responsive.

TOO MANY BOUNDARIES

REACTIVITY

TOO FEW BOUNDARIES

- Performance: app is less responsive.
- You lose focus or scroll position or don’t handle an event if an important DOM node is recomputed.

TOO MANY BOUNDARIES

REACTIVITY

TOO FEW BOUNDARIES

- Performance: app is less responsive.
- You lose focus or scroll position or don’t handle an event if an important DOM node is recomputed.
- Correctness problems, if you start trying to re-render less often or re-render manually.

TOO MANY BOUNDARIES

REACTIVITY

TOO FEW BOUNDARIES

- Performance: app is less responsive.
- You lose focus or scroll position or don’t handle an event if an important DOM node is recomputed.
- Correctness problems, if you start trying to re-render less often or re-render manually.

TOO MANY BOUNDARIES

- Performance: views render slowly initially.

REACTIVITY

TOO FEW BOUNDARIES

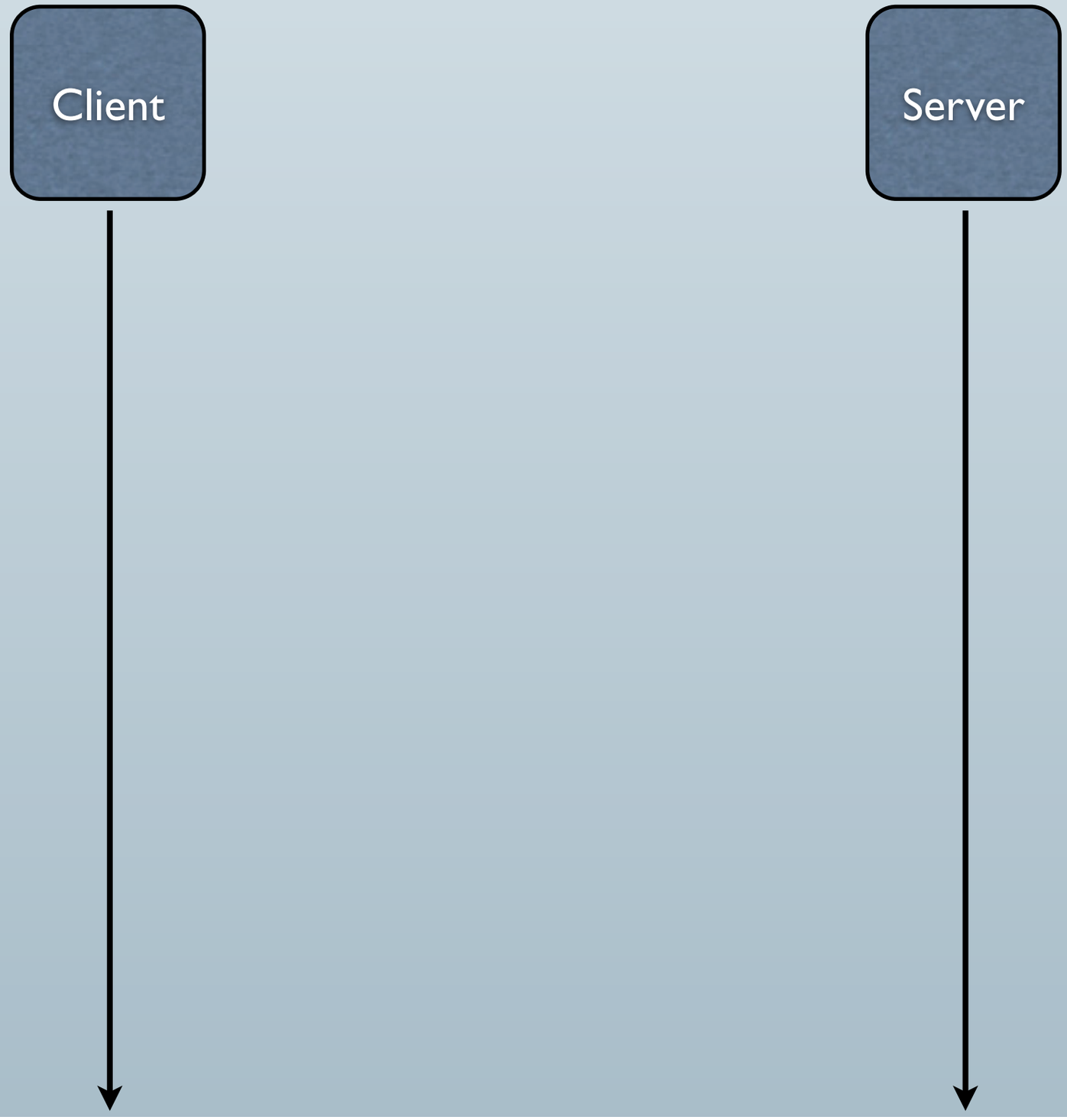
- Performance: app is less responsive.
- You lose focus or scroll position or don’t handle an event if an important DOM node is recomputed.
- Correctness problems, if you start trying to re-render less often or re-render manually.

TOO MANY BOUNDARIES

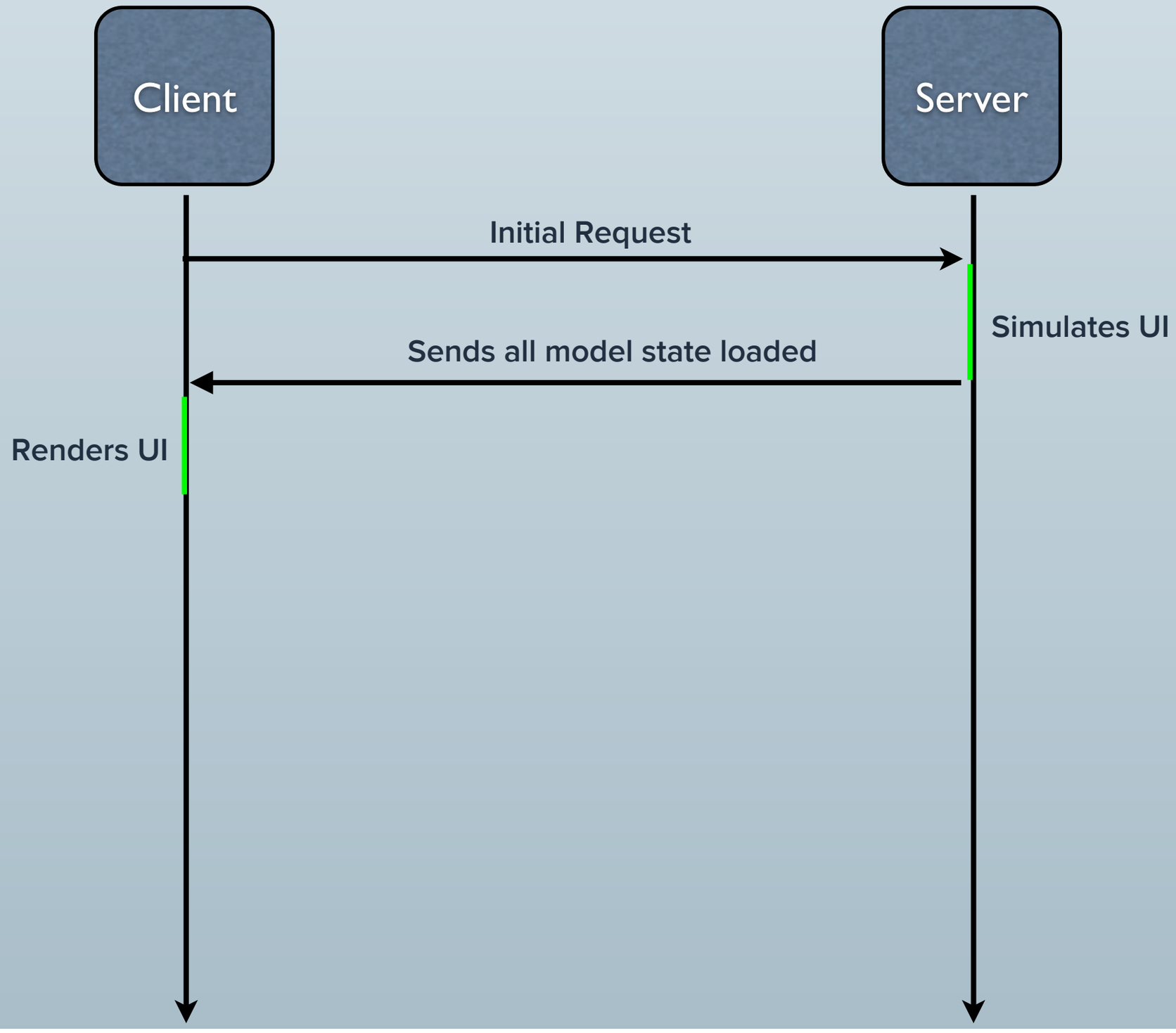
- Performance: views render slowly initially.
- Debugging is near-impossible, because the dependency graph is too complicated.

</REACTIVITY>

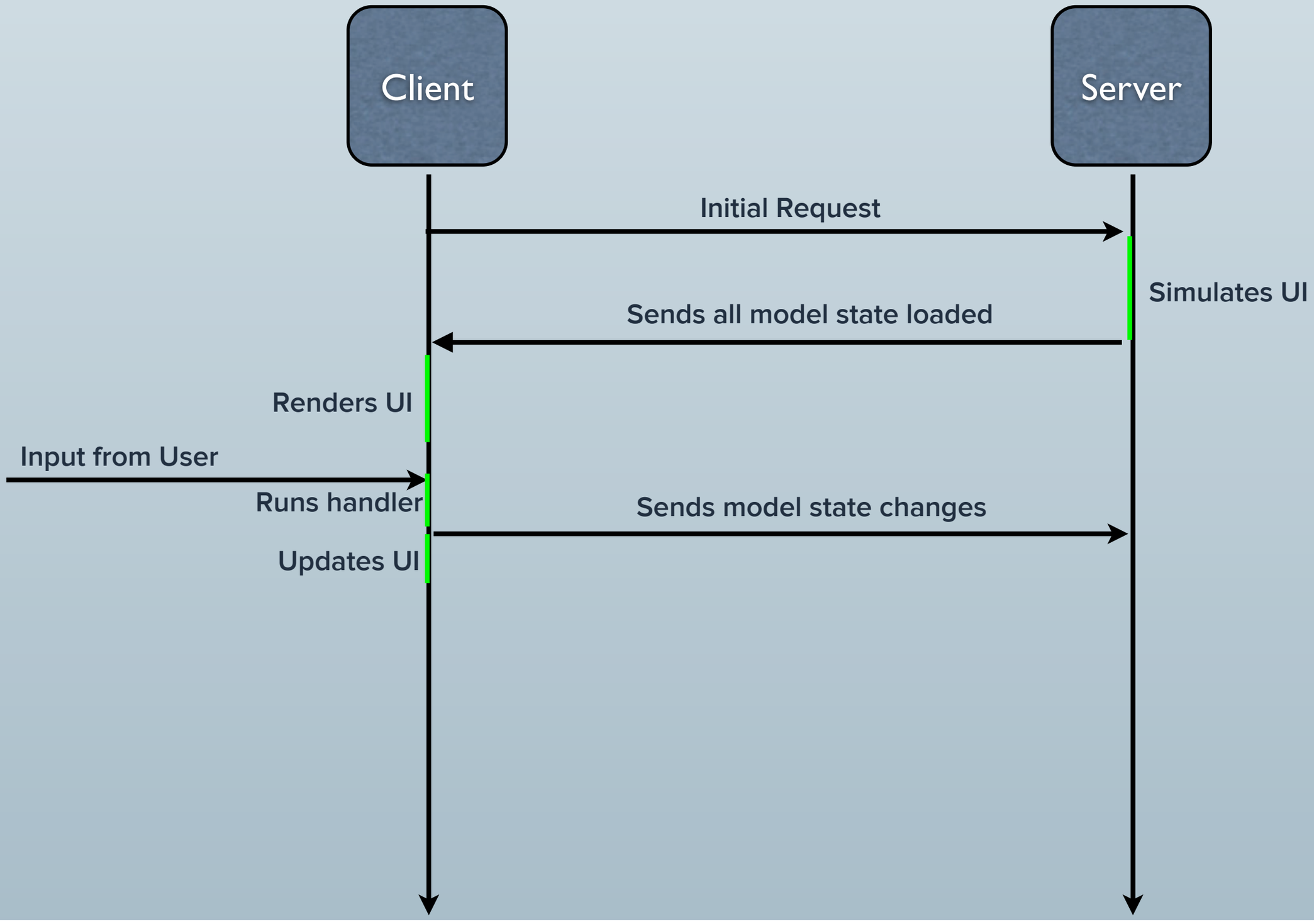
SYNC AND SIMULATION



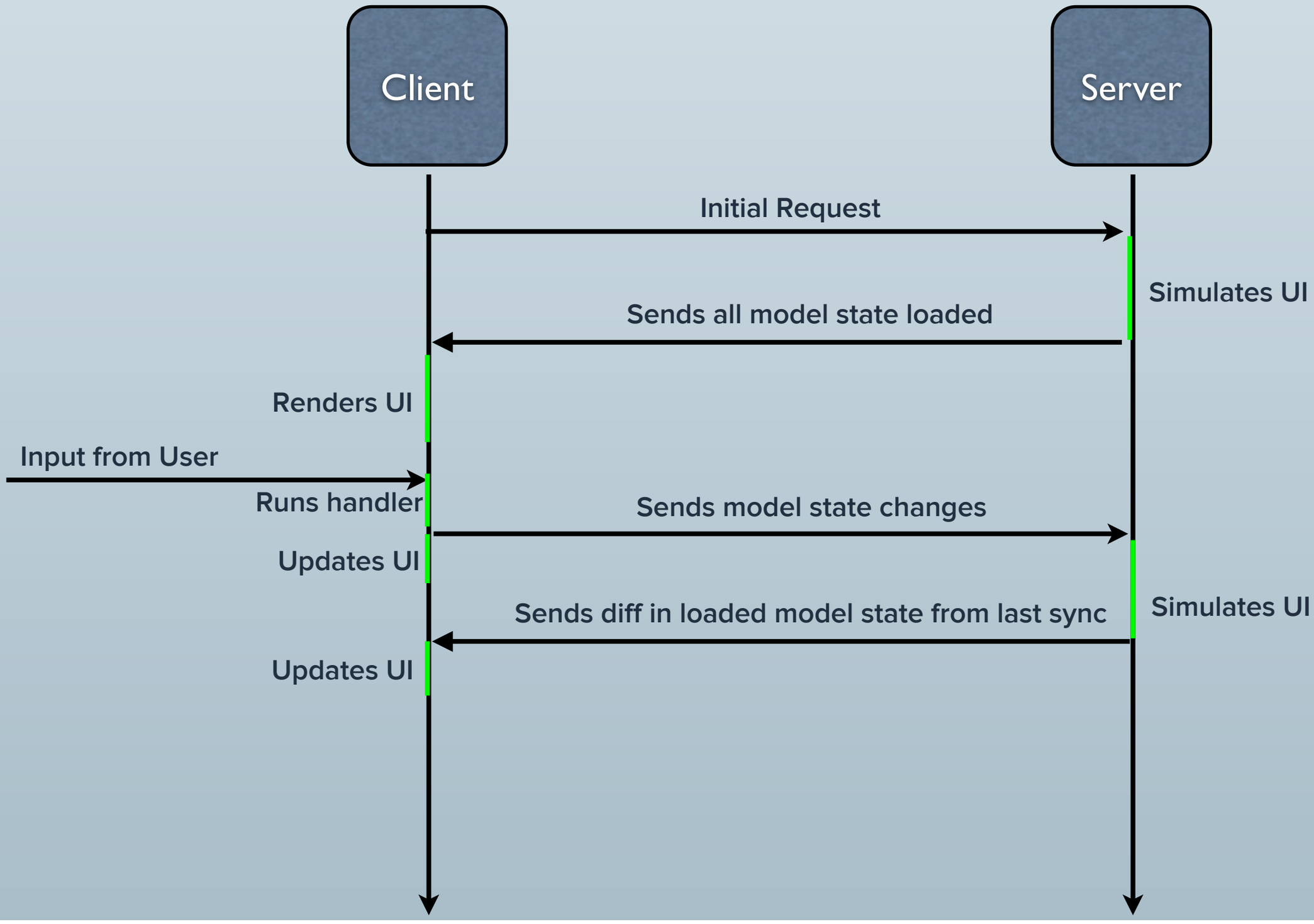
SYNC AND SIMULATION



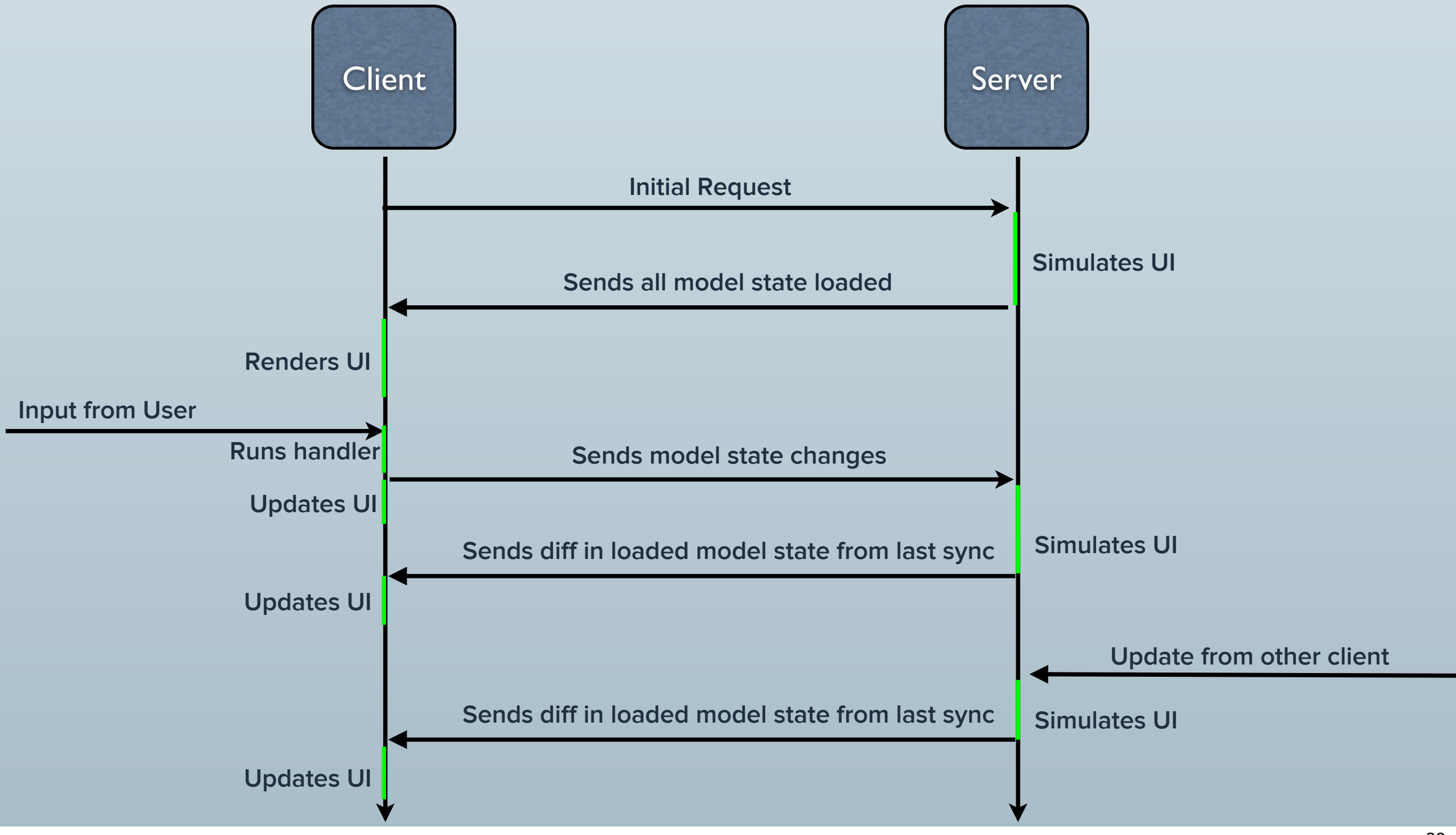
SYNC AND SIMULATION



SYNC AND SIMULATION



SYNC AND SIMULATION



SYNC AND SIMULATION

BENEFITS

REQUIREMENTS

SYNC AND SIMULATION

BENEFITS

- One codebase. Data-fetching code for a view lives within that view.

REQUIREMENTS

SYNC AND SIMULATION

BENEFITS

- One codebase. Data-fetching code for a view lives within that view.

- Eliminates tons of serialization/AJAX code.

REQUIREMENTS

SYNC AND SIMULATION

BENEFITS

- One codebase. Data-fetching code for a view lives within that view.

- Eliminates tons of serialization/AJAX code.

- Huge class of tricky situations abstracted into one well understood system.

REQUIREMENTS

SYNC AND SIMULATION

BENEFITS

- One codebase. Data-fetching code for a view lives within that view.
- Eliminates tons of serialization/AJAX code.
- Huge class of tricky situations abstracted into one well understood system.

REQUIREMENTS

- Short-circuiting for O(n) complex views.

SYNC AND SIMULATION

BENEFITS

- One codebase. Data-fetching code for a view lives within that view.
- Eliminates tons of serialization/AJAX code.
- Huge class of tricky situations abstracted into one well understood system.

REQUIREMENTS

- Short-circuiting for O(n) complex views.
- Automated and/or explicit batching.

SYNC AND SIMULATION

BENEFITS

- One codebase. Data-fetching code for a view lives within that view.
- Eliminates tons of serialization/AJAX code.
- Huge class of tricky situations abstracted into one well understood system.

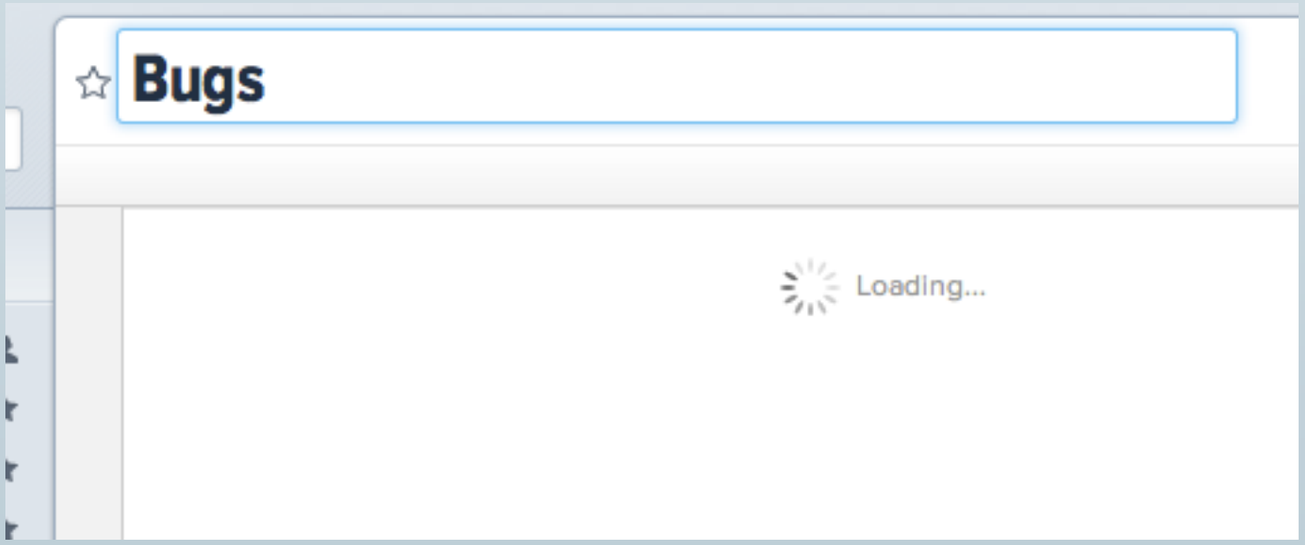
REQUIREMENTS

- Short-circuiting for O(n) complex views.
- Automated and/or explicit batching.
- Memory-intensive stateful web servers and stateful load-balancing.

</SYNC>

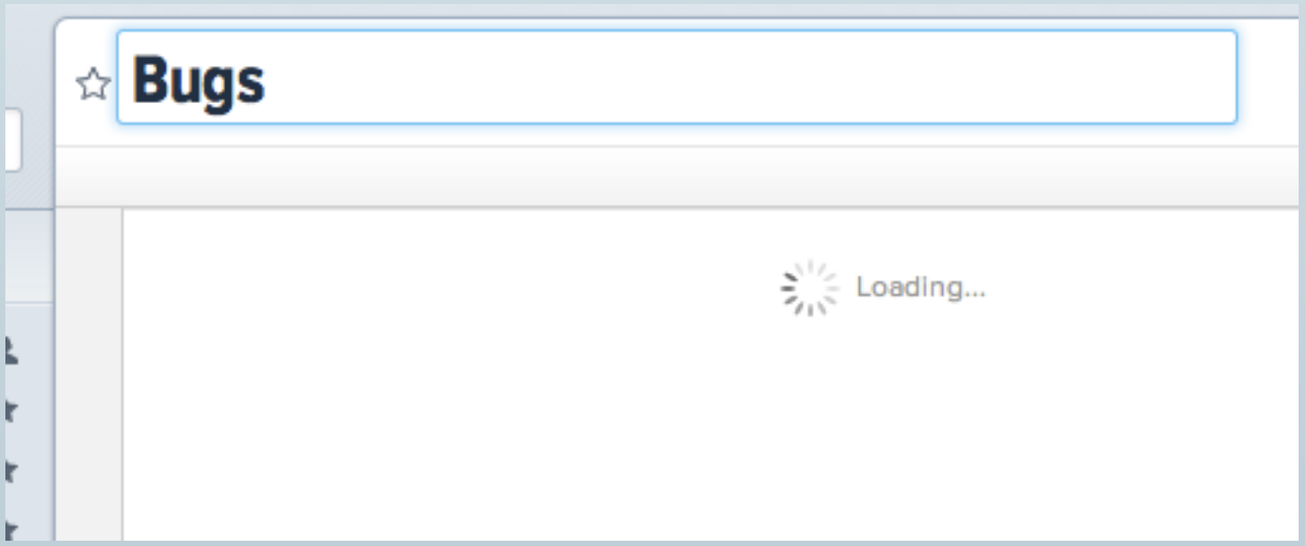
UNKNOWNS

What happens when you switch lists in Asana?



UNKNOWNS

What happens when you switch lists in Asana?

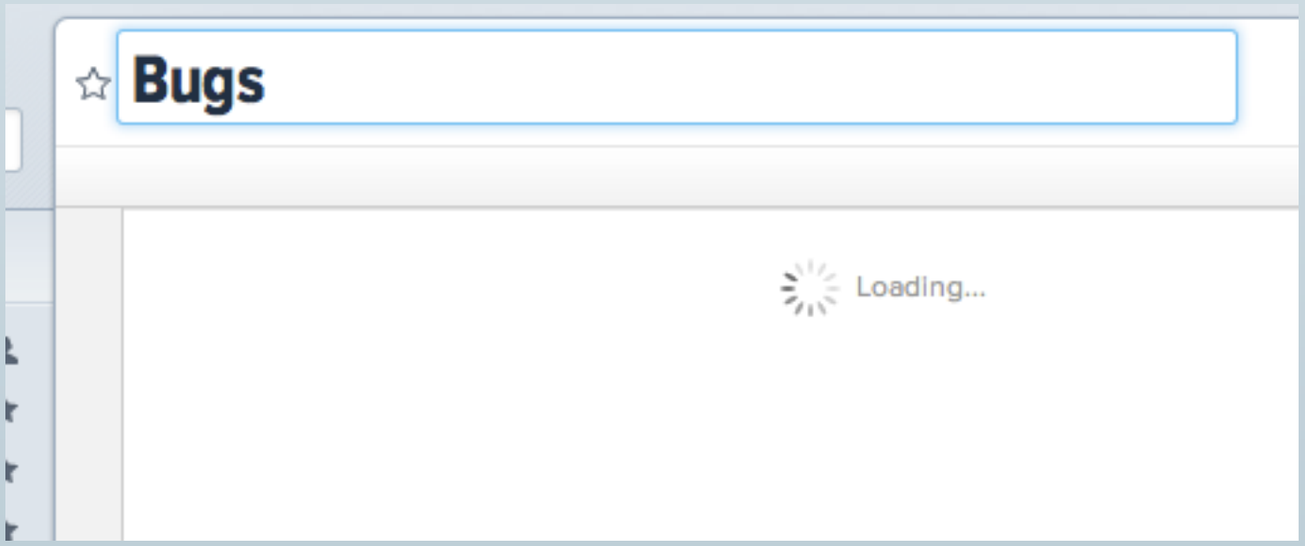


list_navigation_view.js:

```
env.appSession().gridState().setCurrentList(new_list);
```

UNKNOWNS

What happens when you switch lists in Asana?



list_navigation_view.js:

```
env.appSession().gridState().setCurrentList(new_list);
```

grid_pane_view.js:

```
DIV([
  grid_pane.renderHeader(),
  AsanaHelpers.renderWithinLoadingBoundary(grid_pane)
])
```

UNKNOWNS

- Any reactive value that tries to load data that is not there gets the value UnknownValue.Loading

UNKNOWNS

- Any reactive value that tries to load data that is not there gets the value UnknownValue.Loading
- Any reactive value that depends upon another reactive value that is unknown becomes unknown itself

UNKNOWNS

- Any reactive value that tries to load data that is not there gets the value UnknownValue.Loading
- Any reactive value that depends upon another reactive value that is unknown becomes unknown itself
- Use loading boundaries to stop the propagation

UNKNOWNS

- Any reactive value that tries to load data that is not there gets the value UnknownValue.Loading
- Any reactive value that depends upon another reactive value that is unknown becomes unknown itself
- Use loading boundaries to stop the propagation
- Also: UnknownValue.AccessDenied

UNKNOWNS

What happens when you hit an unknown in a handler?

UNKNOWNS

What happens when you hit an unknown in a handler?

You crash!

UNKNOWNS

What happens when you hit an unknown in a handler?
You crash!

How do you prevent this?

Add assertions that run in development that no handler touches any data that wasn't fetched by the view for that handler.

UNKNOWNS

What happens when you hit an unknown in a handler?
You crash!

How do you prevent this?

Add assertions that run in development that no handler touches any data that wasn't fetched by the view for that handler.

OR:

Allow handlers to try to run themselves on the client, but run on the server if they hit an unknown.

</UNKNOWNS>

ONE MORE THING.

Steve Jobs.

QUESTIONS?

- What do you work on?
    - What happened with Lunascript?
- How does Asana compare to your past jobs?
- How is Asana changing as an organization?
    - What are your favorite other tech companies?
        - When is feature X coming to Asana?
- What's the difference between Luna and LiveNode?
    - Are there open source equivalents to Luna?