# Matrix

It’s the Developer eXperience.

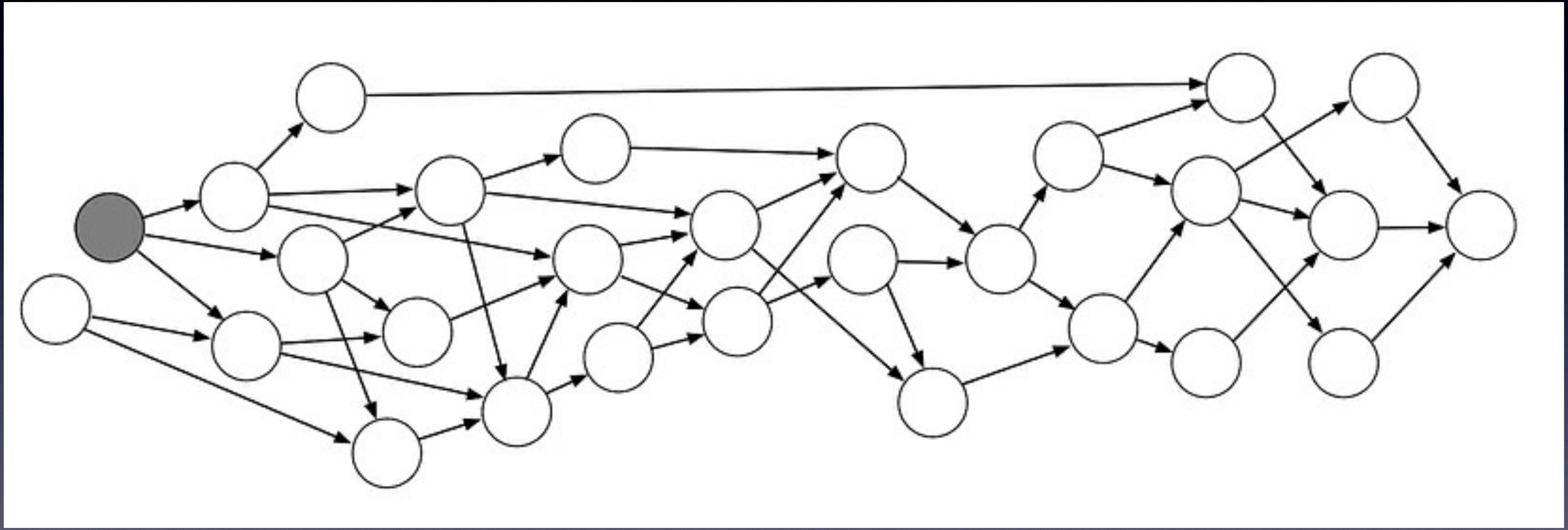
# Standard Reactivity

- State dependency
- State change propagation
- Side effects off change

# Not Standard Reactivity

- Granularity (object properties, not view functions)
- The OO prototype model (ad hoc properties as needed)
- Transparency (no explicit subscribe or notify)
- In-place state management (object properties, view and domain, managed directly)
- *Glitch-free* state propagation
- Unlimited state scope for rules and event handlers
- Extensible to arbitrary non-reactive libraries (XHR, routing, Postgres,…)

# Pratītyasamutpāda

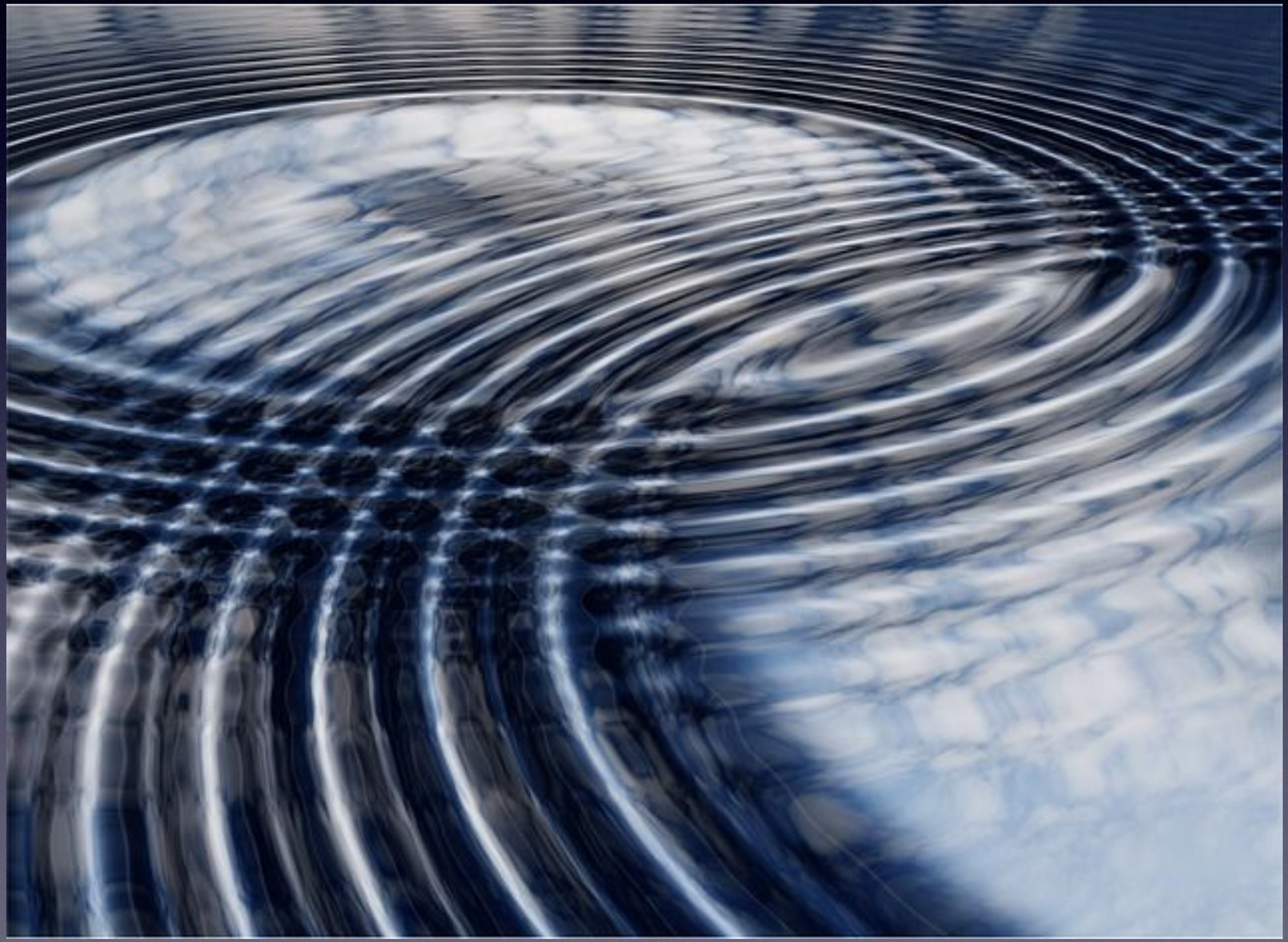


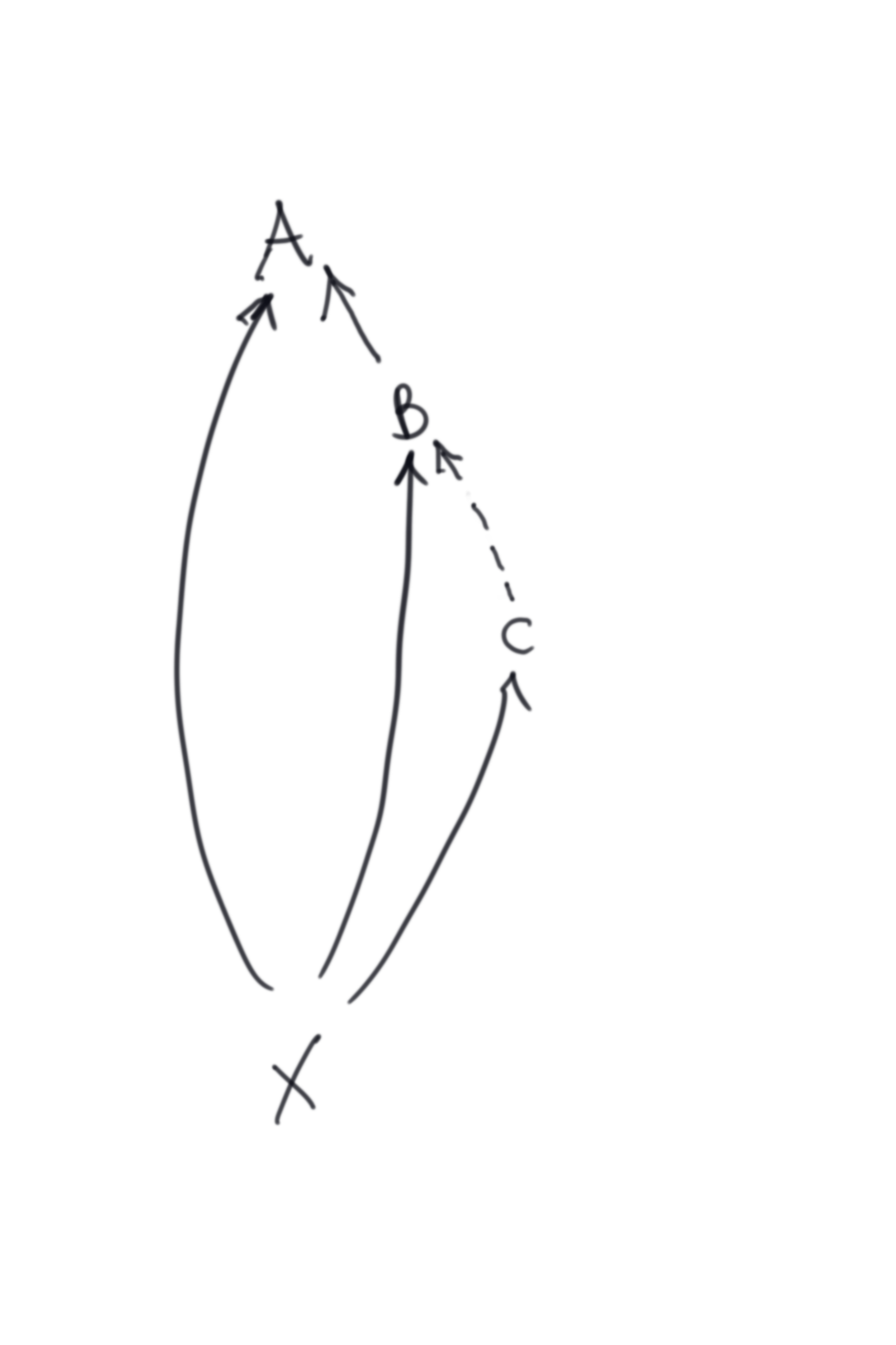
Everything is connected.

# Matrix

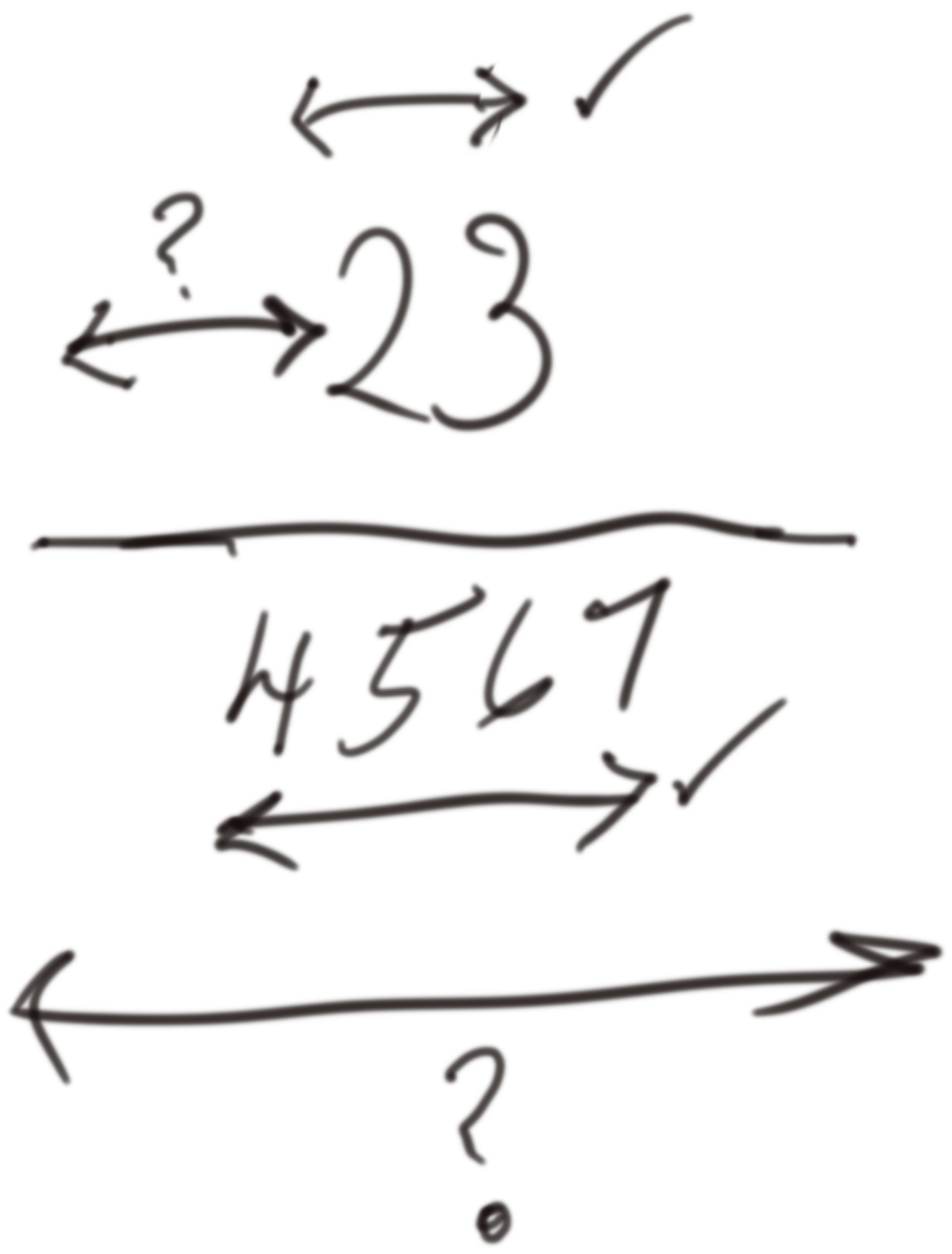
- ✓ Accurate, complete, de-duplicated state propagation
- ✓ Side effects off change, outside the DAG propagation
- ✓ Granularity
- ✓ Transparency
- ✓ In-place state management
- ✓ Glitch-free state propagation
- ✓ Unlimited scope

# Glitches

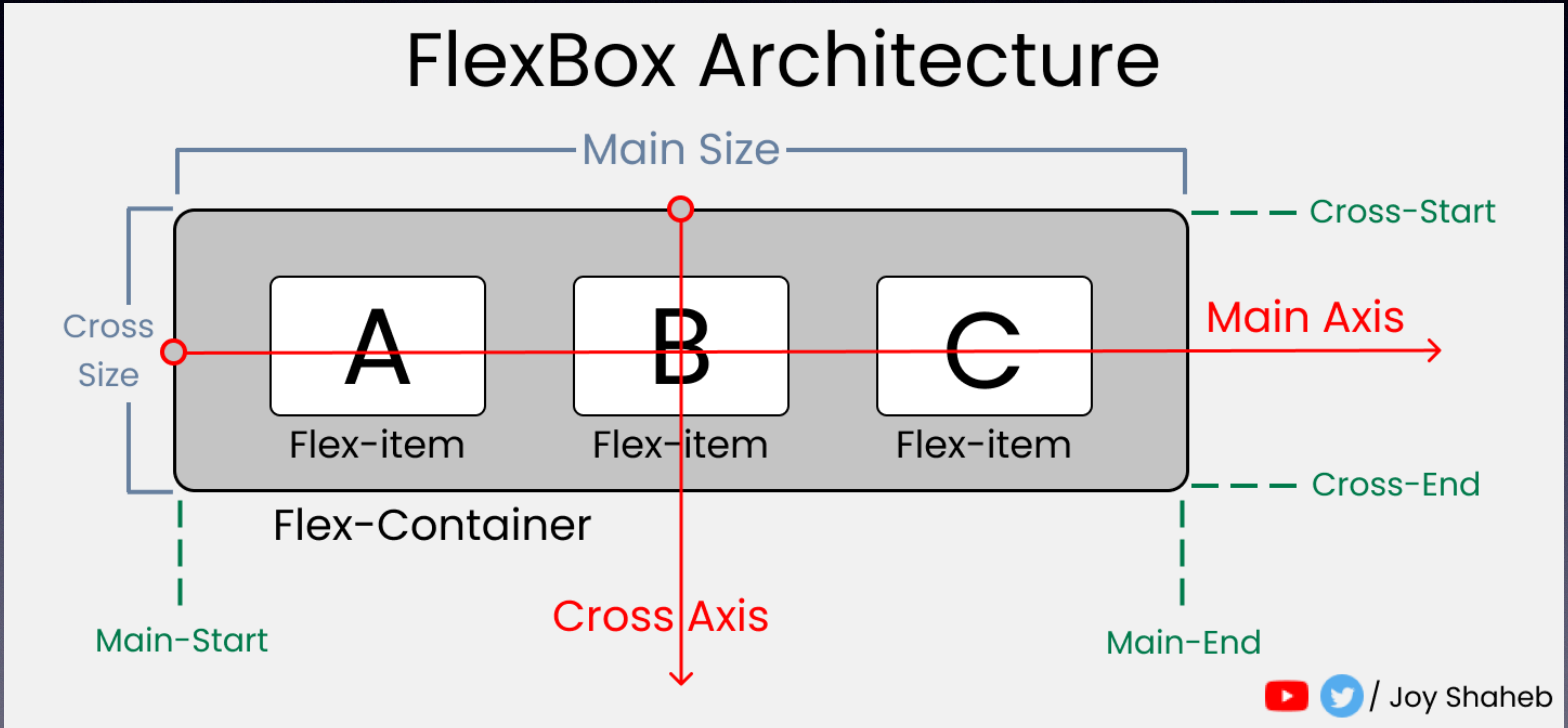




# The Birth of Matrix

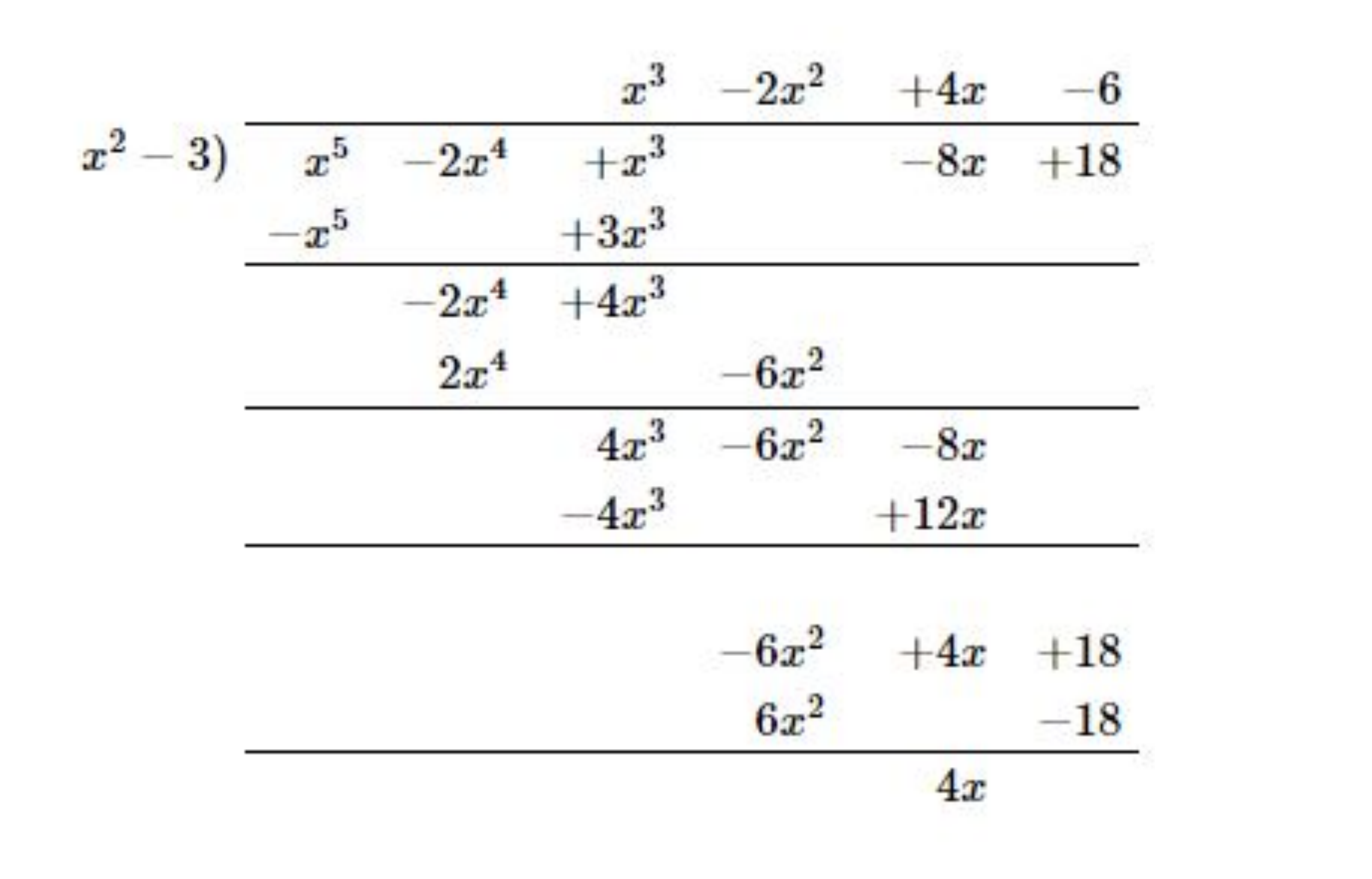


# We did not have Flexbox



# We did not have MathJax

```
<span class="math display">\[\begin{array}{r}
 \\
x^2 - 3) \\
\phantom{.} \\
\phantom{.} \\
\phantom{.} \\
\phantom{.} \\
\phantom{.} \\
\phantom{.} \\
\phantom{.} \\
\phantom{.} \\
\phantom{.} \\
\end{array}
\begin{array}{rrrrrr}
    &amp; &amp; x^3 &amp; -2x^2 &amp; +4x &amp; -6 \\ \hline
 x^5 &amp; -2x^4 &amp; + x^3 &amp; &amp;-8x &amp; +18 \\
-x^5 &amp; &amp;+3x^3 \\ \hline
 &amp; -2x^4 &amp; +4x^3 \\
 &amp; 2x^4 &amp; &amp;-6x^2 \\ \hline
&amp; &amp; 4x^3 &amp; -6x^2 &amp; -8x \\
&amp; &amp; -4x^3 &amp; &amp;+12x \\ \hline \\
&amp; &amp; &amp;-6x^2 &amp; +4x &amp; +18 \\
&amp; &amp; &amp; 6x^2 &amp; &amp; -18 \\ \hline
&amp; &amp; &amp; &amp; 4x
\end{array}\]</span>
```



$$\begin{array}{rrrrrrr}
 & & & x^3 & -2x^2 & +4x & -6 \\ \cline{2-7}
x^2-3) & x^5 & -2x^4 & +x^3 & & -8x & +18 \\
 & -x^5 & & +3x^3 & & & \\ \cline{2-7}
 & & -2x^4 & +4x^3 & & & \\
 & & 2x^4 & & -6x^2 & & \\ \cline{2-7}
 & & & 4x^3 & -6x^2 & -8x & \\
 & & & -4x^3 & & +12x & \\ \cline{2-7}
 & & & & & & \\
 & & & & -6x^2 & +4x & +18 \\
 & & & & 6x^2 & & -18 \\ \cline{2-7}
 & & & & & 4x & 
\end{array}$$

# My failed scheme:

## Compute children then parent

N-offset-hz = (F-width - N-width)/2

But F-width = max( N-right, D-right)

And N-right = N-offset-hz + string-width( N.value)

Oops. Hello, Cycle

“Make everything as simple as possible, but no simpler.”
— Albert Einstein

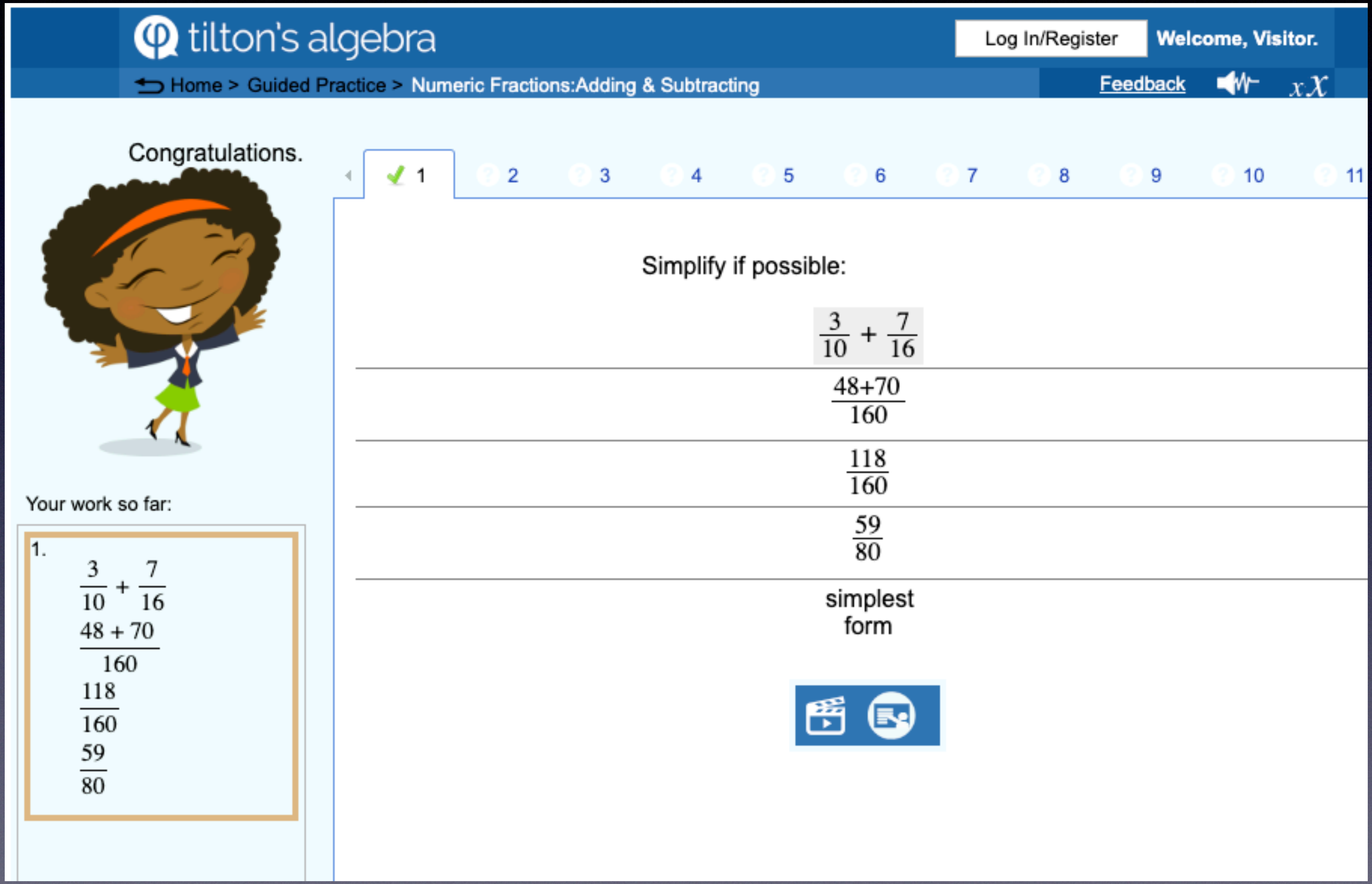
# Hello, Matrix

```
N.left = 0
N.right = string-width( N.value )
D.right = string-width( D.value )
F.width = max( N-right, D-right )
N.offset-h = (F.width - N.right)/2
```

# Fraction geometry

```
(make-mx
   :type :ratio
   :width (c-rule
             (apply max (map width (parts me))))
   :parts (c-rule
             (make-mx ;; numerator
                :value (c-input nil)
                :left 0
                :right (c-rule (string-width (value me)))
                :offset-hz (c-rule (/ (- (width parent)
                                         (width me)) 2)))
             (make-mx ;; denominator
                  …similar))
```

# Three years later…



Congratulations.

Your work so far:

1. $\frac{3}{10}+\frac{7}{16}$

$\frac{48+70}{160}$

$\frac{118}{160}$

$\frac{59}{80}$

Simplify if possible:

$\frac{3}{10}+\frac{7}{16}$

$\frac{48+70}{160}$

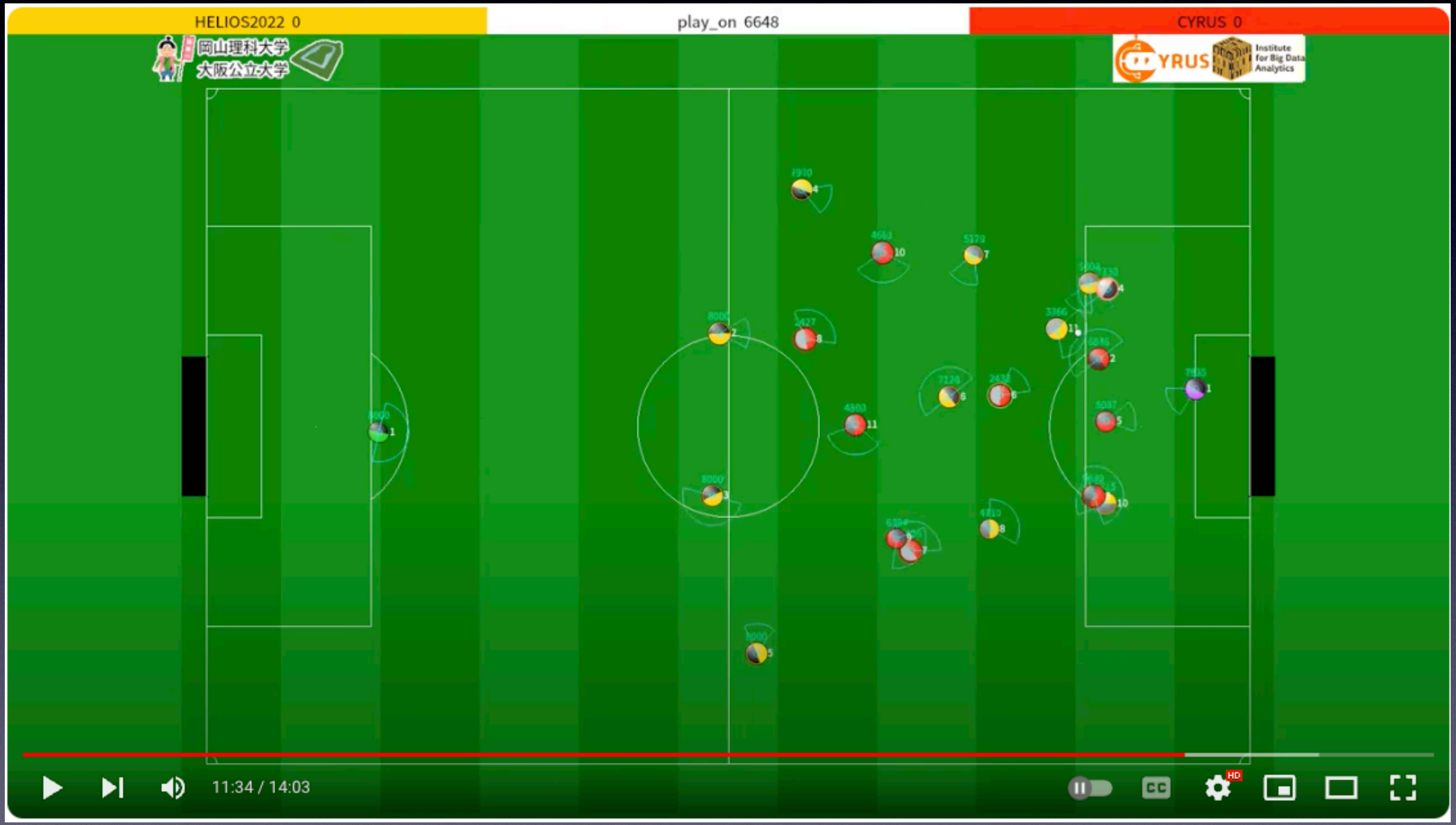
$\frac{118}{160}$

$\frac{59}{80}$

simplest form

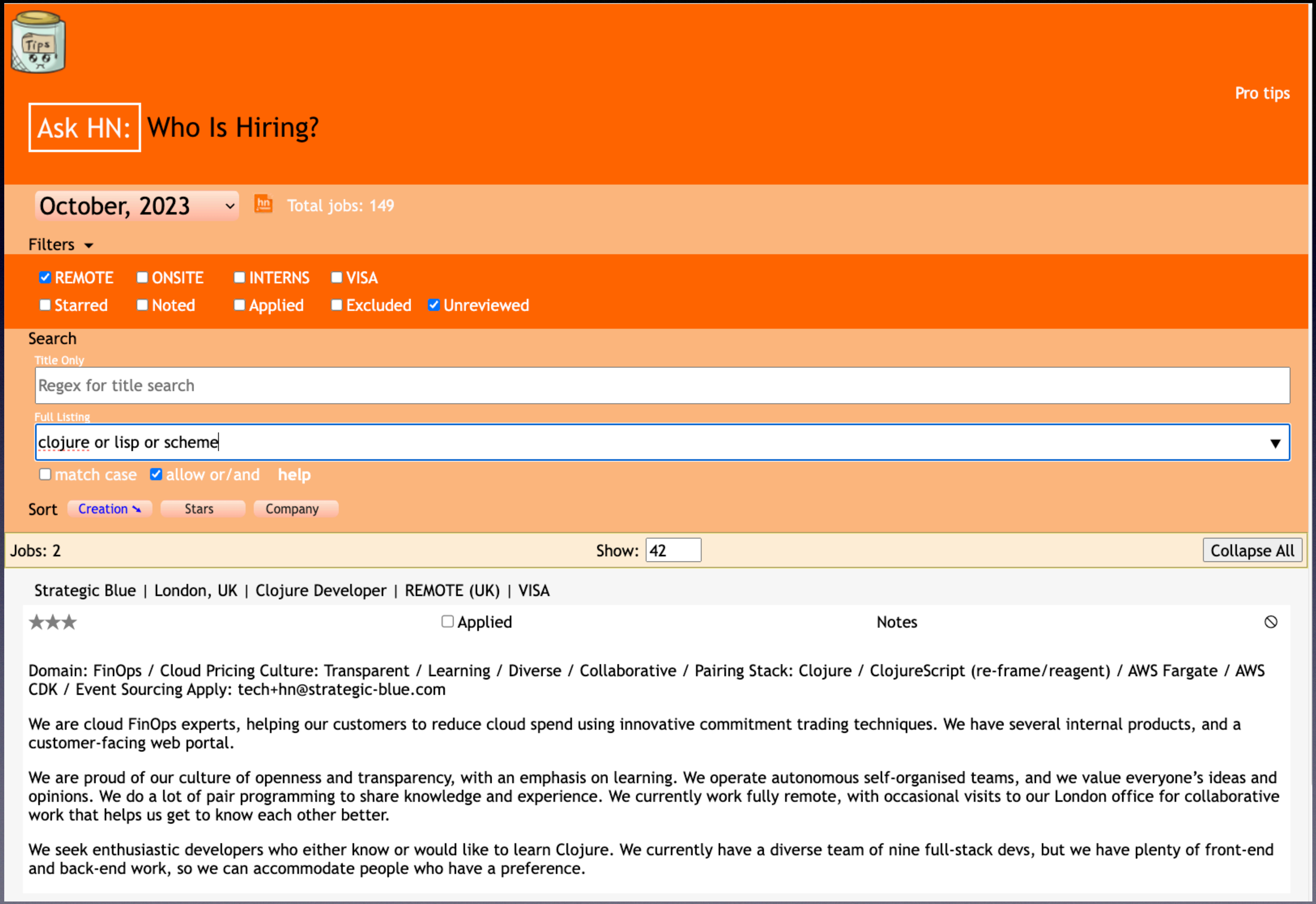
http://tiltonsalgebra.com/#

# Not Just the UI



https://www.robocup.org/leagues/23

# AskHN Who's Hiring Browser



Pro tips

Ask HN: Who Is Hiring?

October, 2023 Total jobs: 149

Filters

☑ REMOTE ☐ ONSITE ☐ INTERNS ☐ VISA
☐ Starred ☐ Noted ☐ Applied ☐ Excluded ☑ Unreviewed

Search

Title Only: Regex for title search

Full Listing: clojure or lisp or scheme

☐ match case ☑ allow or/and help

Sort: Creation ↘ | Stars | Company

Jobs: 2 Show: 42 Collapse All

Strategic Blue | London, UK | Clojure Developer | REMOTE (UK) | VISA

★★★ ☐ Applied Notes

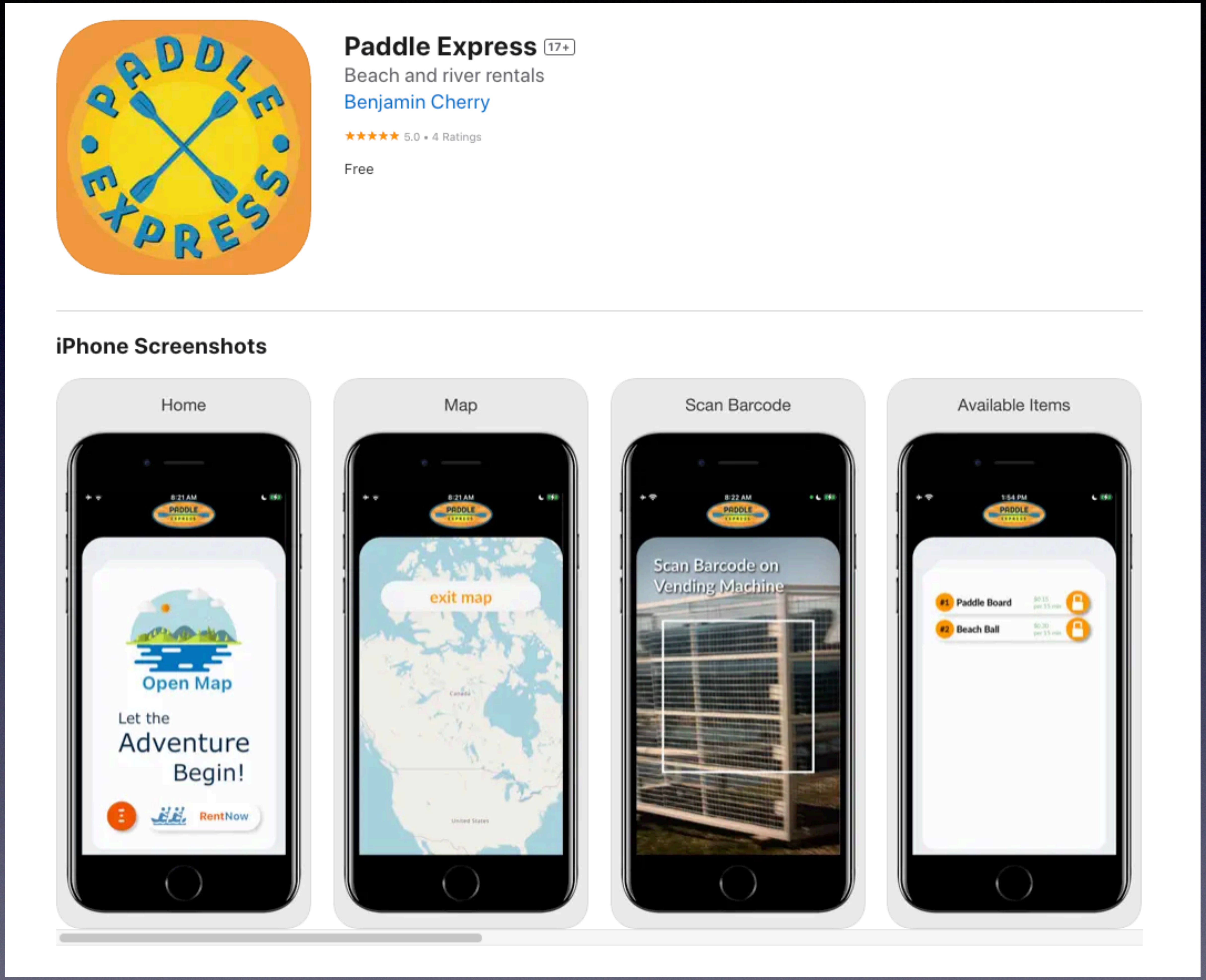
Domain: FinOps / Cloud Pricing Culture: Transparent / Learning / Diverse / Collaborative / Pairing Stack: Clojure / ClojureScript (re-frame/reagent) / AWS Fargate / AWS CDK / Event Sourcing Apply: tech+hn@strategic-blue.com

We are cloud FinOps experts, helping our customers to reduce cloud spend using innovative commitment trading techniques. We have several internal products, and a customer-facing web portal.

We are proud of our culture of openness and transparency, with an emphasis on learning. We operate autonomous self-organised teams, and we value everyone's ideas and opinions. We do a lot of pair programming to share knowledge and experience. We currently work fully remote, with occasional visits to our London office for collaborative work that helps us get to know each other better.

We seek enthusiastic developers who either know or would like to learn Clojure. We currently have a diverse team of nine full-stack devs, but we have plenty of front-end and back-end work, so we can accommodate people who have a preference.

# Paddle Express



**Paddle Express** 17+

Beach and river rentals

Benjamin Cherry

★★★★★ 5.0 • 4 Ratings

Free

**iPhone Screenshots**

A Flutter/MX App

# Back to Properties

```
(defn td-completed [todo]
 (mget todo :completed))

(defn clear-completed-button []
  (fx/visibility
    {:visible (cF (when-let [tds (todo/app-todos (my-app))]
                    (some td-completed tds)))}
    (fx/text-button
      {:onPressed (mswap! (mget (fasc :app) :todo-list) :kids
                    (partial remove td-completed))}
      (fx/text "Clear done"))))
```

# Reactivity and Persistence

```
(deftype ToDo []
  :extends Model
  PObserver
  (observe [this prop me new-value prior-value cell]
    (storage/td-rewrite
       (select-keys @td [:stg-id :created-at :title :completed])))))
```

OOP is not all bad.

# The OO Prototype Model

```
(defn todo-list-item [todo]
  (fx/visibility
    {:visible (cF (mget me :selected?))}
    {:selected? (cF (case (fmu :todo-routing :selection)
                      :all true
                      :active (not (td-completed todo))
                      :done (td-completed todo)))}
    (fx/container
      (fx/gesture-detector
        {:onDoubleTap (as-dart-callback []
                        (mset! me :editing? true))}
        {:name :item-control
         :editing? (c-input false)}
        (cF ;; hello responsive UI: widget children vary appropriately
           (if (mget me :editing?)
              (to-do-editor me todo)
              (to-do-display todo)))))))
```

We can add custom properties in a second map as needed.

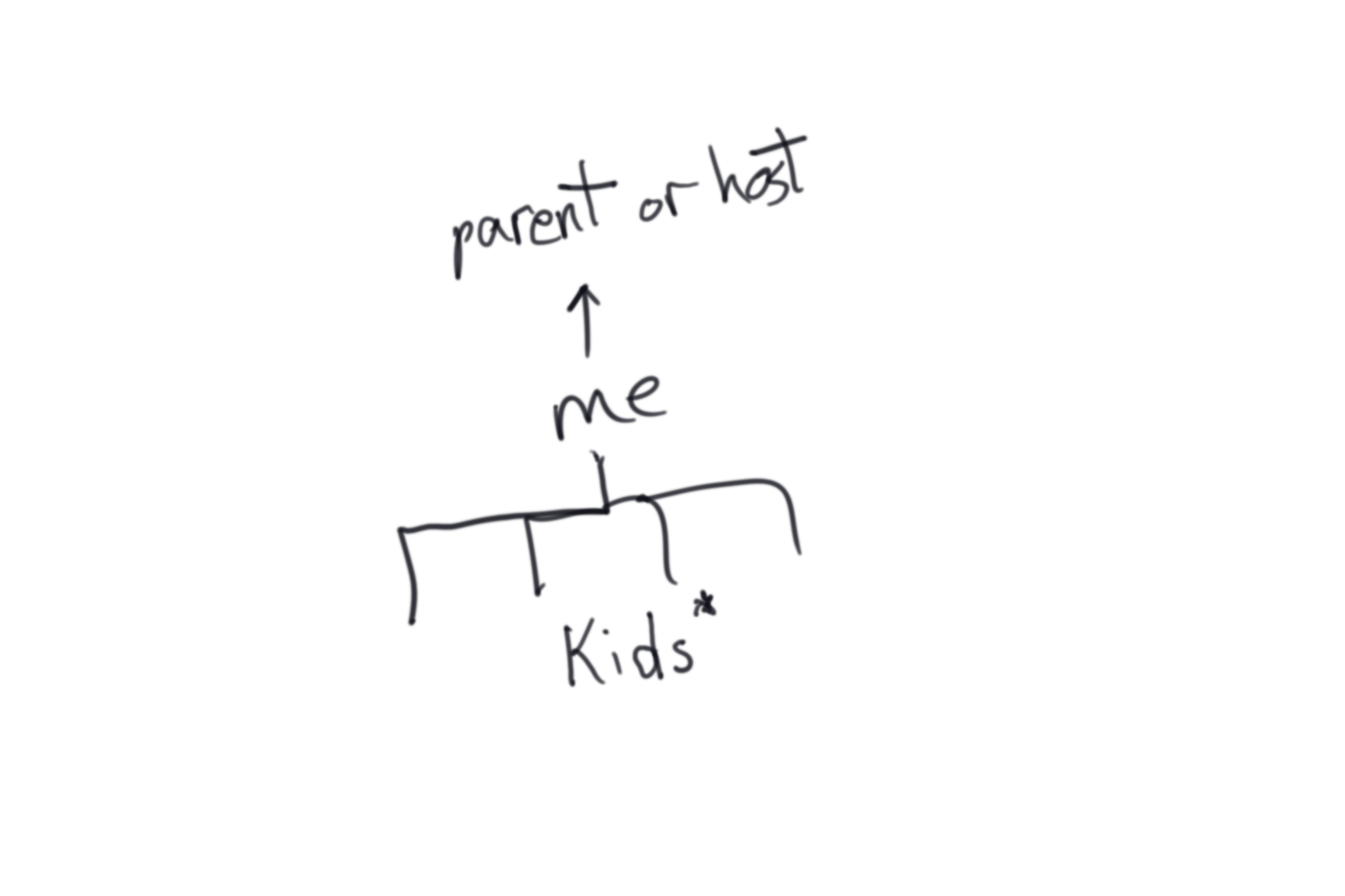
# Navigation

Utilities `fasc` and `fm*` search the DAG
From one model to any other so we can act on it.

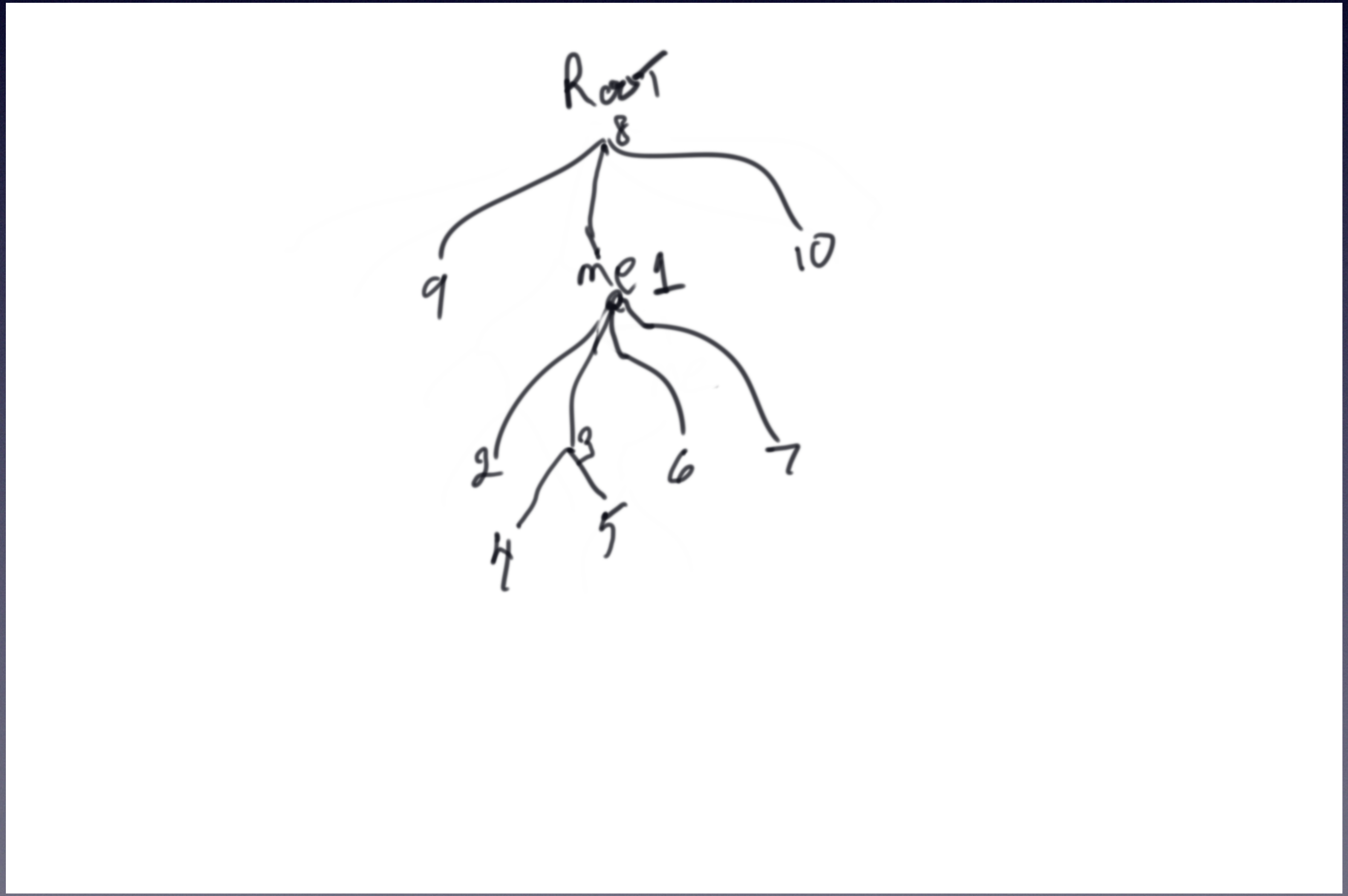
```
(defn to-do-editor [me todo]
  (fx/list-tile
    {:title (cF (fx/focus-scope
                  {:onFocusChange
                     (cF (as-dart-callback [focused?]
                           (when (not focused?)
                             (commit-to-do-editing (fm* :todo-editor) todo)
                             (mset! (fasc :item-control) :editing? false))))}
                  (fx/text-field {}
                    {:name  :todo-editor
                     :value (cI title)})))))}))
```

# Navigation and Natural Scope

Each model



A tree of models



# How it works (simplified)

```
(defn cell-get [c]
  (let [prior-value (c-value c)
        new-value (ensure-value-is-current c :c-read nil)]
   (when *depender*
      (record-dependency c))
    new-value))
```

```
(defn compute-cell [c]
  (when (some #{c} *call-stack*)
    (mx-throw “COMPUTE_CYCLE_DETECTED"))
  (binding [*depender* c
            *call-stack* (cons c *call-stack*)]
    (unlink-from-used c)
    ((c-formula c) c))))
```

```
(defn cell-change [c new-value]
  (let [prior-value (c-value c)]
    (cell-pulse-bump c)
    (md-prop-value-store (c-model c) (c-prop c) new-value)
    (propagate-to-callers c callers)
    (cell-watch c prior-value)))
```

# Transparency

- Subscriptions: transparent and automatic
- Notify: transparent and automatic
- Navigation:  modest hinting a good idea; definite risk of cycles. Hinting avoids cycles. Otherwise, just say what we seek.

Moral: we can have our one-way, acyclic DAG and ignore it, too.

## Side effects

Show code for triggering setState, for updating TTS parameters, for writing to Hive. Prolly one slide each.

# When I Use Matrix, and When Not

Matrix helps build any application involving:

- an interesting amount…
- …of long-lived state maintained over time; and
- a stream of unpredictable inputs.

Examples:

- a human working on Algebra problems in  a UI
- A bot playing RoboCup with a UDP game server

Counter-example:

- ETL/ELT: each record handled in isolation.

# Matrix Evolving

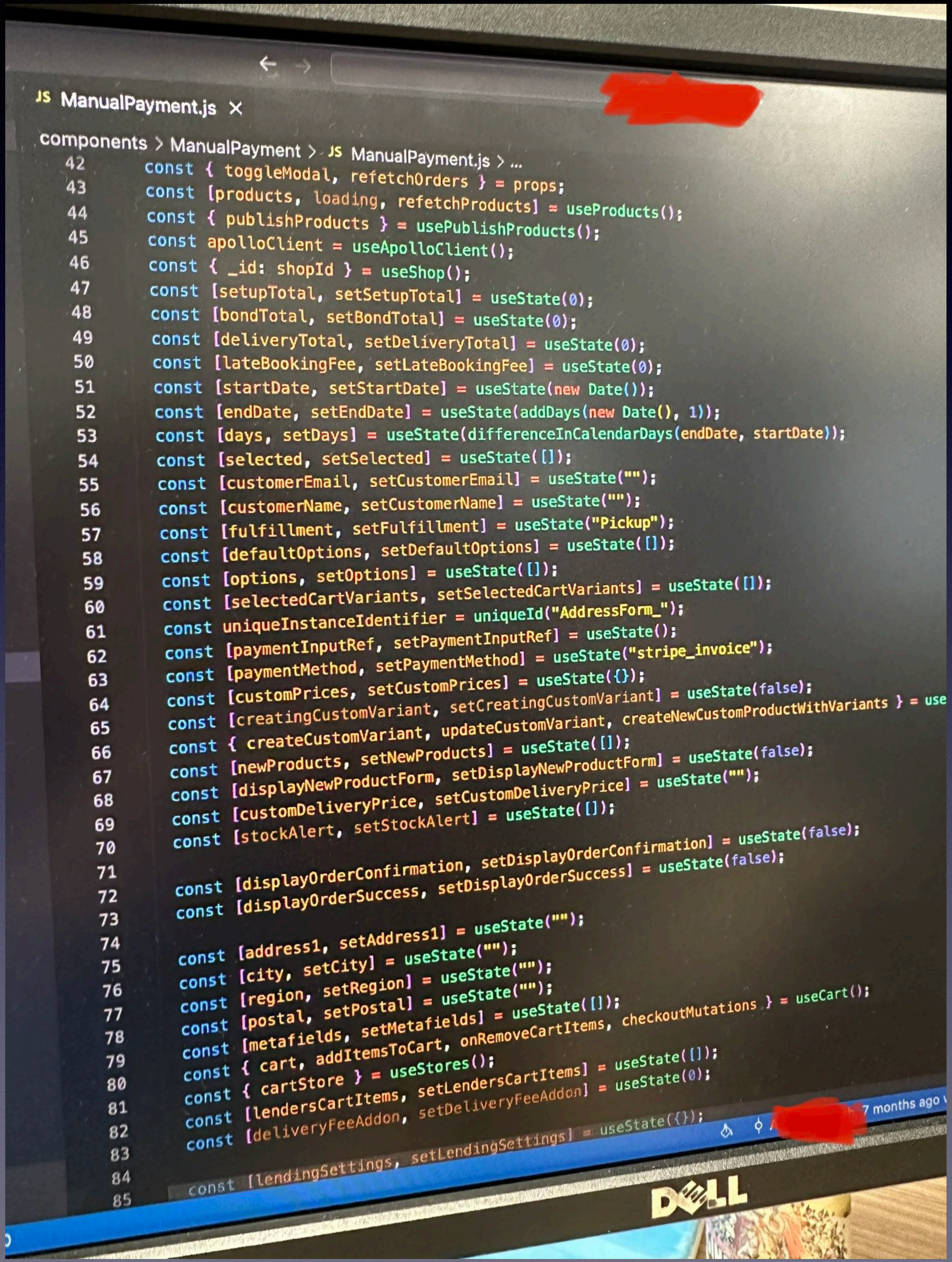
Every new project brings new ideas.

- Flutter's propensity for async led to the `:async?` option.
- This summer a stream analyzing project led to a cell "freeze" capability.
- Preparing this talk led to a relaxation of a rule enjoining mutation of a formula.
- An app using a physics engine led to dependencies on hash-table entities.

# There is more

- Formulas become inputs.
- Formulas become constants.
- Lazy cells.
- Synapses.
- Cells deciding to freeze.
- Without-dependency,
- Unchanged-if.
- Standalone cells.
- Client queue handler.

# Where Flux Went Wrong



JS ManualPayment.js ×

[redacted]

components > ManualPayment > JS ManualPayment.js > ...

```
    const { toggleModal, refetchOrders } = props;
    const [products, loading, refetchProducts] = useProducts();
    const { publishProducts } = usePublishProducts();
    const apolloClient = useApolloClient();
    const { _id: shopId } = useShop();
    const [setupTotal, setSetupTotal] = useState(0);
    const [bondTotal, setBondTotal] = useState(0);
    const [deliveryTotal, setDeliveryTotal] = useState(0);
    const [lateBookingFee, setLateBookingFee] = useState(0);
    const [startDate, setStartDate] = useState(new Date());
    const [endDate, setEndDate] = useState(addDays(new Date(), 1));
    const [days, setDays] = useState(differenceInCalendarDays(endDate, startDate));
    const [selected, setSelected] = useState([]);
    const [customerEmail, setCustomerEmail] = useState("");
    const [customerName, setCustomerName] = useState("");
    const [fulfillment, setFulfillment] = useState("Pickup");
    const [defaultOptions, setDefaultOptions] = useState([]);
    const [options, setOptions] = useState([]);
    const [selectedCartVariants, setSelectedCartVariants] = useState([]);
    const uniqueInstanceIdentifier = uniqueId("AddressForm_");
    const [paymentInputRef, setPaymentInputRef] = useState();
    const [paymentMethod, setPaymentMethod] = useState("stripe_invoice");
    const [customPrices, setCustomPrices] = useState({});
    const [creatingCustomVariant, setCreatingCustomVariant] = useState(false);
    const { createCustomVariant, updateCustomVariant, createNewCustomProductWithVariants } = use
    const [newProducts, setNewProducts] = useState([]);
    const [displayNewProductForm, setDisplayNewProductForm] = useState(false);
    const [customDeliveryPrice, setCustomDeliveryPrice] = useState("");
    const [stockAlert, setStockAlert] = useState([]);

    const [displayOrderConfirmation, setDisplayOrderConfirmation] = useState(false);
    const [displayOrderSuccess, setDisplayOrderSuccess] = useState(false);

    const [address1, setAddress1] = useState("");
    const [city, setCity] = useState("");
    const [region, setRegion] = useState("");
    const [postal, setPostal] = useState("");
    const [metafields, setMetafields] = useState([]);
    const { cart, addItemsToCart, onRemoveCartItems, checkoutMutations } = useCart();
    const { cartStore } = useStores();
    const [lendersCartItems, setLendersCartItems] = useState([]);
    const [deliveryFeeAddon, setDeliveryFeeAddon] = useState(0);
    const [lendingSettings, setLendingSettings] = useState({});
```

[redacted] 7 months ago

"Just one more state variable bro. Just one more hook and the page will have everything it needs. Just one more state variable please bro. Bro? Add one more state variable please bro"

This makes me wanna go back to MVC and separation of concerns. Where did we go wrong.

— Heard on Twitter Oct. 2023

# Summary: the Matrix Difference

| | |
|---|---|
| Granularity | Read and write domain/view *properties* |
| Transparency | No explicit subscribe or notify; dependencies identified automatically |
| In-place state management | Direct maintenance of domain and view properties; no separate store |
| Navigation 1 | Formulas can read any app state |
| Navigation 2 | Event handlers can modify any app state |
| Extensibility | Readily applied to external libraries |