AWESOME STATE MANAGEMENT FOR REACT*

*AND OTHER VIRTUAL-DOM LIBRARIES

Fred Daoud - @foxdonut00

WHY AWESOME?

NOT A LIBRARY. A SIMPLE PATTERN.

WORKS WITH ANY VIRTUAL DOM LIBRARY

VIRTUAL DOM IS NICE

```
VIEW = FUNCTION(MODEL)
```

“ HOW DO I MANAGE STATE? ”

REDUX

MOBX

CEREBRAL

CYCLE.JS

LET'S LOOK AT

A DIFFERENT APPROACH.

MEIOSIS

HTTP://MEIOSIS.JS.ORG

THE MEIOSIS PATTERN

- model ← single source of truth
- view = function(model)
- update model
- → view is automatically re-rendered

- model = single source of truth

```
const initialModel = { counter: 0 };
```

- view = function(model)

```
const view = model => (
  <div>Counter is {model.counter}</div>
);
```

- actions update the model

```
const createActions = update => ({
  increment: () => update(model => {
    model.counter++;
    return model;
  })
  // after calling update(...),
  // view is automatically re-rendered
});
```

- views call actions

```
const createView = actions => model => (
  <div>
    <div>Counter is {model.counter}</div>
    <button onClick={actions.increment}>Increment</button>
  </div>
);
```

WHAT IS **UPDATE**?

HOW DO WE AUTOMATICALLY RE-RENDER THE VIEW?

MEIOSIS PATTERN IMPLEMENTATION

- Minimal streams: just `map` and `scan`
- Or, write your own minimal implementation

FLYD STREAMS

MAP

```
const s1 = flyd.stream();
const s2 = s1.map(value => value * 10);
s2.map(value => console.log(value));

s1(5);
s1(10);

// console output is
50
100
```

FLYD STREAMS

SCAN

```
const s1 = flyd.stream();
const add = (x, y) => x + y;
const s2 = flyd.scan(add, 0, s1);
s2.map(value => console.log(value));

s1(5); s1(13); s1(24);

// console output is
0 5 18 42
```

THE MEIOSIS PATTERN

```
const initialModel = { counter: 0 };
const update = flyd.stream();
const applyUpdate = (model, modelUpdate) => modelUpdate(model);
const models = flyd.scan(applyUpdate, initialModel, update);

const view = createView(createActions(update));
const element = document.getElementById("app");
models.map(model => ReactDOM.render(view(model), element));
```

USING DIFFERENT VIRTUAL DOM LIBS

```
models.map(model => ReactDOM.render(view(model), element));
models.map(model => Inferno.render(view(model), element));
models.map(model => m.render(element, view(model)));
models.map(model => preact.render(view(model), element,
  element.lastElementChild));

const render = view => element = patch(element, view);
models.map(model => render(view(model)));
```

MODEL UPDATE

PLAIN

```
update(model => {
  model.counter++;
  return model;
});
```

LODASH

```
update(model => _.update(model, "counter", _.partial(_.add,1)));
```

LODASH FP

```
update(_.update("counter", _.add(1)));
```

RAMDA

```
update(R.over(R.lensProp("counter"), R.add(1)));
```

IMMUTABLE.JS

```
update(model => model.update("counter", v => v + 1));
```

COMPONENTS

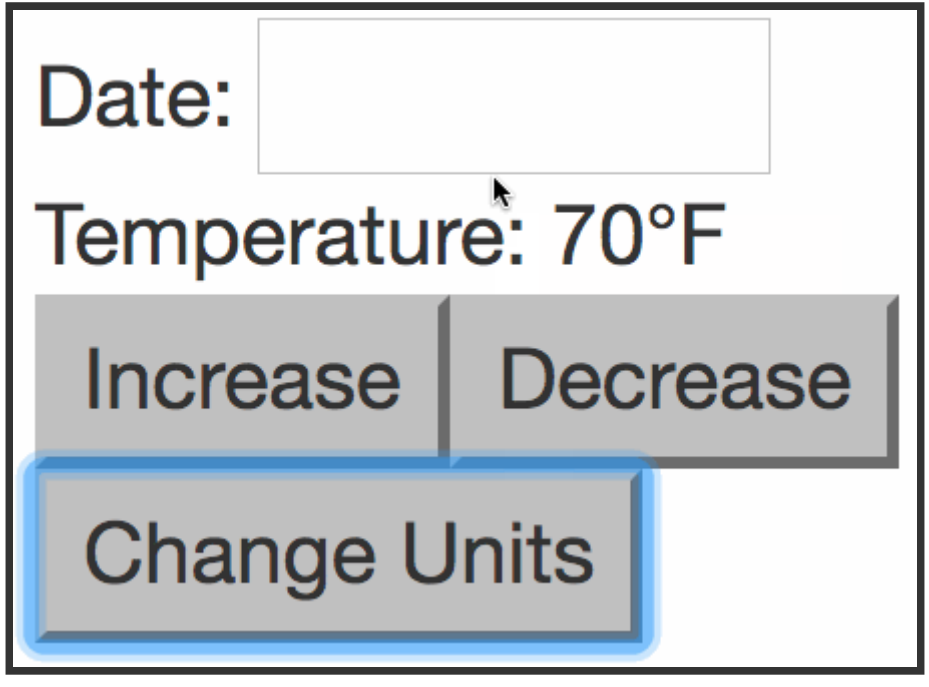
A COMPONENT IS JUST AN OBJECT WITH FUNCTIONS

```
import { createActions } from "./actions";
import { createView } from "./view";

// This is the same 'update' stream ↓↓
export const createTemperature = update => ({
  model: () => ({
    date: "",
    value: 20,
    units: "C"
  }),

  view: createView(createActions(update))
});
```

TEMPERATURE EXAMPLE



ACTIONS (1/2)

```
export const createActions = update => ({
  editDate: evt =>
    update(model => {
      model.date = evt.target.value;
      return model;
    }),

  increase: amount => () =>
    update(model => {
      model.value = model.value + amount;
      return model;
    }),
```

ACTIONS (2/2)

```
  changeUnits: () => update(model => {
    if (model.units === "C") {
      model.units = "F";
      model.value = Math.round( model.value * 9 / 5 + 32 );
    }
    else {
      model.units = "C";
      model.value = Math.round( (model.value - 32) / 9 * 5 );
    }
    return model;
  })
});
```

VIEW

```
export const createView = actions => model => (
  <div>
    <div>Date: <input type="text" size="10" value={model.date}
      onChange={actions.editDate}/></div>
    <span>Temperature: {model.value}&deg;{model.units} </span>
    <div>
      <button onClick={actions.increase(1)}>Increase</button>
      <button onClick={actions.increase(-1)}>Decrease</button>
    </div>
    <div>
      <button onClick={actions.changeUnits}>Units</button>
    </div>
  </div>
);
```

THE MEIOSIS PATTERN

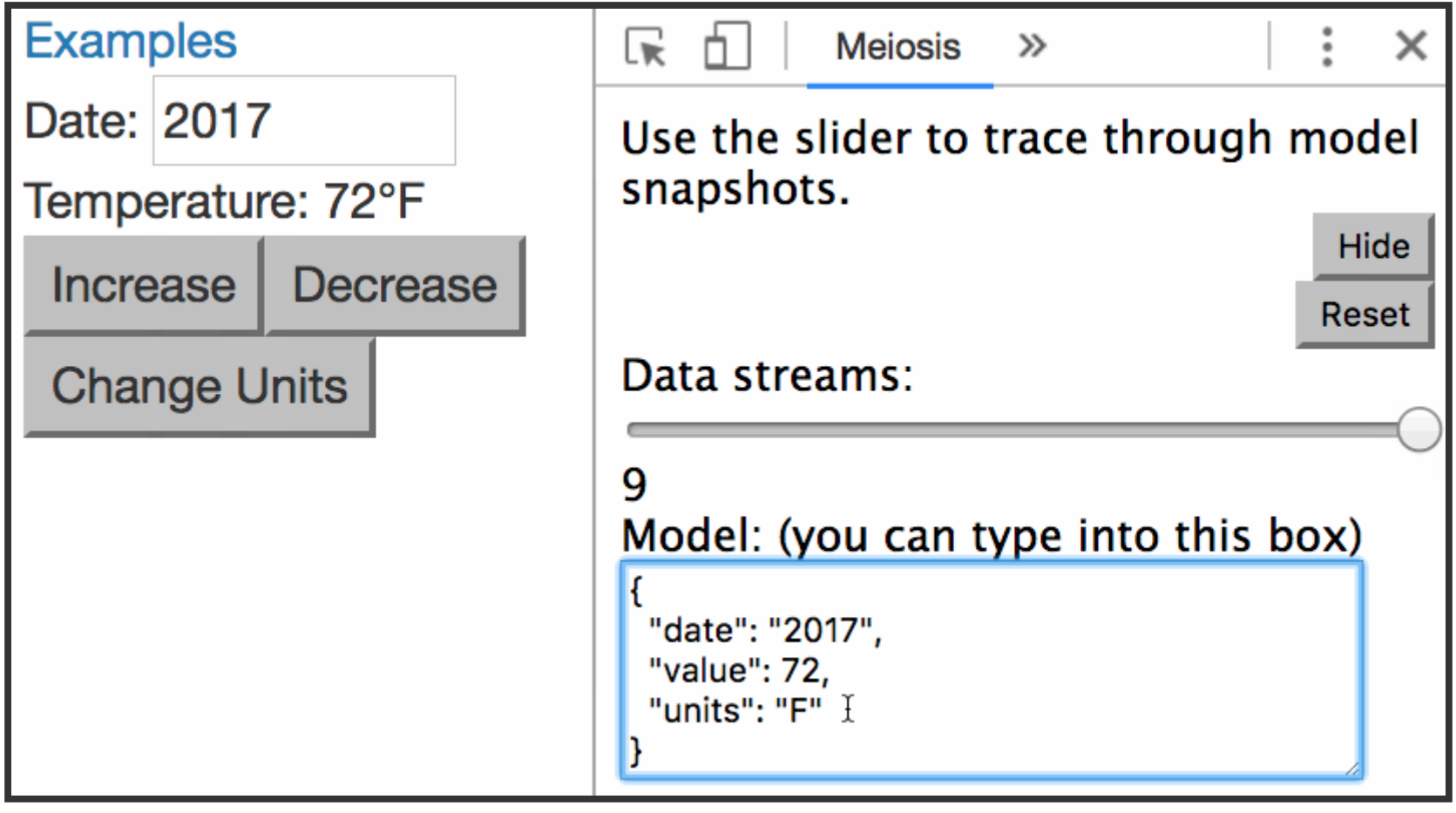
```
const update = flyd.stream();
const temperature = createTemperature(update); // <--------
const initialModel = temperature.model();      // <--------
const applyUpdate = (model, modelUpdate) => modelUpdate(model);
const models = flyd.scan(applyUpdate, initialModel, update);

const element = document.getElementById("app");
models.map(model =>
  ReactDOM.render(temperature.view(model), // <--------
    element));
```

MEIOSIS TRACER

TIME-TRAVEL DEV TOOL

MEIOSIS TRACER IN CHROME DEVTOOLS



MEIOSIS TRACER IN PAGE

Examples

Date: 2017-09

Temperature: 72°F

Hide

Reset

Increase | Decrease

Change Units

Data streams:

8

Model: (you can type into this box)


```
{
  "date": "2017-09",
  "value": 72,
  "units": "F"
}
```

USING MEIOSIS TRACER IN CHROME

- Install `meiosis` as a dev dependency
- Install the Chrome extension

```
// Only for using Meiosis Tracer in development.
import { trace } from "meiosis";
trace({ update, dataStreams: [ models ] });
```

USING MEIOSIS TRACER IN PAGE

- Also install `meiosis-tracer` as a dev dep.

```
<div id="tracer"></div>

// Only for using Meiosis Tracer in development.
import { trace } from "meiosis";
import meiosisTracer from "meiosis-tracer";
trace({ update, dataStreams: [ models ] });
meiosisTracer({ selector: "#tracer" });
```

REUSABLE COMPONENTS

REUSING THE TEMPERATURE COMPONENT

```
export const createApp = update => {
  const components = {
    air: createTemperature(nest(update, "air")),
    water: createTemperature(nest(update, "water"))
  };

  return {
    model: () => ({
      air: components.air.model(),
      water: components.water.model()
    }),
    view: createView(components)
  };
};
```

NESTING THE **UPDATE** FUNCTION

```
export const nest = (update, path) =>
  modelUpdate => update(model => {
    model[path] = modelUpdate(model[path]);
    return model;
  });
```

THE VIEW

```
export const createView = components => model => (
  <div>
    <h4>App</h4>
    {components.air.view(model.air)}
    {components.water.view(model.water)}
  </div>
);
```

REUSING THE TEMPERATURE COMPONENT

App

Date: 2017-0

Temperature: 73°F

Increase | Decrease

Change Units

Date:

Temperature: 18°C

Increase | Decrease

Change Units

Use the slider to trace through model snapshots.

Hide

Reset

Data streams:

12

Model: (you can type into this box)


```
{
  "air": {
    "date": "2017-0",
    "value": 73,
    "units": "F"
  },
  "water": {
    "date": "",
    "value": 18,
    "units": "C"
  }
}
```

PATH REPETITION

```
const components = {
  air: createTemperature(nest(update, "air")),
  water: createTemperature(nest(update, "water"))
};

model: () => ({
  air: components.air.model(),
  water: components.air.model()
})

<div>
  {components.air.view(model.air)}
  {components.water.view(model)}
</div>
```

ELIMINATING PATH REPETITION

```
const components = createComponents(update, {
  air: createTemperature,   // passes nest(update, "air")
  water: createTemperature  // also wraps view(model["water"])
});

return {
  // returns { air: c.air.model(), water: c.water.model() }
  model: combineComponents(components, "model"),
  view: createView(components)
};

<div>
  {components.air.view(model)}
  {components.water.view(model)}
</div>
```

COMPUTED PROPERTIES

COMPUTED PROPERTIES (1/2)

```
export const createTemperature = update => {
  const computed = model => {
    let temp = model.value;

    if (model.units === "F") {
      temp = Math.round((temp - 32) * 5 / 9);
    }
    model.comment = (temp < 10) ? "COLD!" :
                    (temp > 40) ? "HOT" : "";

    return model;
  };
```

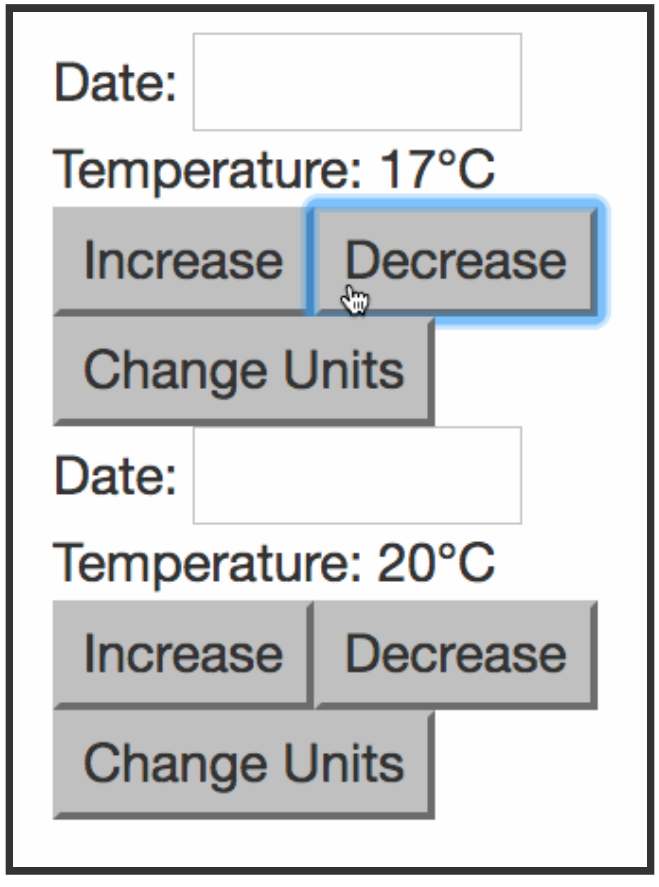
COMPUTED PROPERTIES (2/2)

```
  const view = createView(createActions(update));

  return {
    model: () => ({
      date: "",
      value: 20,
      units: "C"
    }),

    view: model => view(computed(model))
    // view: R.compose(view, computed)
  };
};
```

COMPUTED PROPERTIES



ROUTING

ROUTING EXAMPLE

ROUTING: PAGES

```
export const pages = {
  home: { id: "Home", tab: "Home" },
  coffee: { id: "Coffee", tab: "Coffee" },
  beer: { id: "Beer", tab: "Beer" },
  beerDetails: { id: "BeerDetails", tab: "Beer" }
};
```

ROUTING: NAVIGATION

```
export const createNavigation = update => {
  const navigate = (page, params = {}) =>
    update(model => Object.assign(model, ({ page, params })));

  const navigateToBeer = () => {
    services.loadBeer().then(beerList => {
      update(model => Object.assign(model, { beerList }));
      navigate(pages.beer);
    });
  };
  return { navigateToHome, navigateToBeer, ... };
};
```

ROUTING: APP

```
export const createApp = (update, navigation) => {
  const homeComponent = createHome(update); //more...
  const pageMap = {
    [pages.home.id]: homeComponent, //more...
  };
  return {
    view: model => {
      const component = pageMap[model.page.id];
      return (
        // render tabs, model.page.tab determines active tab
        {component.view(model)}
      );
    }
  };
};
```

ROUTING: ROUTES

```
export const createRouter = navigation => {
  const routes = {
    "/": { id: pages.home.id,
      action: navigation.navigateToHome },
    "/coffee/:id?": { id: pages.coffee.id,
      action: navigation.navigateToCoffee },
    "/beer": { id: pages.beer.id,
      action: navigation.navigateToBeer },
    "/beer/:id": { id: pages.beerDetails.id,
      action: navigation.navigateToBeerDetails }
  };
```

ROUTING: RESOLVE ROUTE

```
import Mapper from "url-mapper";

const resolveRoute = () => {
  const route = document.location.hash.substring(1);
  const resolved = urlMapper.map(route, routes);
  if (resolved) {
    resolved.match.action(resolved.values);
  }
};

window.onpopstate = resolveRoute;
```

ROUTING: ROUTE SYNC

```
const routeMap = Object.keys(routes).reduce((result, route) => {
  result[routes[route].id] = route;
  return result;
}, {});

const routeSync = model => {
  const segment = routeMap[model.page.id] || "/";
  const route = urlMapper.stringify(segment, model.params||{});
  if (document.location.hash.substring(1) !== route) {
    window.history.pushState({}, "", "#" + route);
  }
};
```

ROUTING EXAMPLE

MEIOSIS

HTTP://MEIOSIS.JS.ORG

- Documentation • Examples
- Tracer • Gitter chat

Fred Daoud • @foxdonut00