My home runs Redux

By Morten Olsen

I have been playing around with smart homes for a long time; I have used most of the platforms out there, I have developed quite a few myself, and one thing I keep coming back to is Redux.

Those who know what Redux is may find this a weird choice, but for those who don't know Redux, I'll give a brief introduction to get up to speed.

Redux is a state management framework, initially built for a React talk by Dan Abramov and is still primarily associated with managing React applications. Redux has a declarative state derived through a "reducer"-function. This reducer function takes in the current state and an event, and, based on that event, it gives back an updated state. So you have an initial state inside Redux, and then you dispatch events into it, each getting the current state and updating it. That means that the resulting state will always be the same given the same set of events.

So why is a framework primarily used to keep track of application state for React-based frontends a good fit for a smart home? Well, your smart home platform most likely closely mimics this architecture already!

First, an event goes in, such as a motion sensor triggering, or you set the bathroom light to 75

...But that is not quite what happens on most platforms. Deterministic events may go into the system, but this usually doesn't cause a change to a deterministic state. Instead, it gets dispatched to the device, the devices updates, the platform sees this change, and then it updates its state to represent that new state.

This distinction is essential because it comes with a few drawbacks:

- Because the event does not change the state but sends a request to the device that does it, everything becomes asynchronous and can happen out of order. This behaviour can be seen either as an issue or a feature, but it does make integrating with it a lot harder from a technical point of view.
- The request is sent to the device as a "fire-and-forget" event. It then relies on the success of that request and the subsequent state change to be reported back from the device before the state gets updated. This behaviour means that if this request fails (something you often see with ZigBee-based devices), the device and the state don't get updated.
- Since the device is responsible for reporting the state change, you are dependent on having that actual device there to make the change. Without sending the changes to the actual device, you cannot test the setup.

So can we create a setup that gets away from these issues?

Another thing to add here is more terminology/philosophy, but most smart home setups are, in my opinion, not really smart, just connected and, to some extent, automated. I want a design that has some actual smartness to it. In this article, I will outline a setup closer to that of the connected, automated home, and at the end, I will give some thoughts on how to take this to the next level and make it smart.

We know what we want to achieve, and Redux can help us solve this. Remember that Redux takes actions and applies them in a deterministic way to produce a deterministic state.

Time to go a bit further down the React rabbit hole because another thing from React-land comes in handy here: the concept of reconciliation.

Instead of dispatching events to the devices waiting for them to update and report their state back, we can rely on reconciliation to update our device. For example, let's say we have a device state for our living room light that says it is at 80

Instead of sending this event to the device, we update the Redux state.

We have a state listener that detects when the state changes and compares it to the state of the actual device. In our case, it seems that the state indicates that the living room light should be at 20

We can also do schedule reconciliation to compare our Redux state to that of the actual devices. If a device fails

to update its state after a change, it will automatically get updated on our next scheduled run, ensuring that our smart home devices always reflect our state.

Sidenote: Yes, of course, I have done a proof of concept using React with a home build reconciliation that reflected the virtual dom unto physical devices, just to have had a house that ran React-Redux

Let's go through our list of issues with how most platforms handle this. We can see that we have eliminated all of them by switching to this Redux-reconciliation approach: we update the state directly to run it synchronously. We can re-run the reconciliation so failed or dropped device updates get re-run. We don't require any physical devices as our state is directly updated.

We now have a robust, reliable, state management mechanism for our smart home, time to add some smarts to it. It is a little outside the article's main focus as this is just my way of doing it; there may be way better ways, so use it at your discretion.

Redux has the concept of middlewares which are stateful functions that live between the event going into Redux and the reducer updating the state. These middlewares allow Redux to deal with side effects and do event transformations.

Time for another piece of my smart home philosophy: Most smart homes act on events, and I have used the word throughout this article, but to me, events are not the most valuable thing when creating a smart home, instead I would argue that the goal is to deal with intents rather than events. For instance, an event could be that I started to play a video on the TV. But, that state a fact, what we want to do is instead capture what I am trying to achieve, the "intent", so lets split this event into two intents; if the video is less than one hour, I want to watch a TV show, if it is more I want to watch a movie.

These intents allow us to not deal with weak-meaning events to do complex operations but instead split our concern into two separate concepts: intent classification and intent execution.

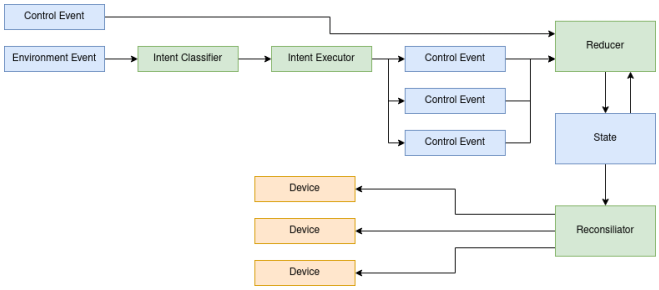
So last thing we need is a direct way of updating devices, as we can not capture everything through our intent classifier. For instance, if I sit down to read a book that does not generate any sensor data for our system to react to, I will still need a way to adjust device states manually. (I could add a button that would dispatch a reading intent)

I have separated the events going into Redux into two types:

- control events, which directly controls a device
- environment events represent sensor data coming in (push on a button, motion sensor triggering, TV playing, etc.)

Now comes the part I have feared, where I need to draw a diagram.

...sorry



So this shows our final setup.

Events go into our Redux setup, either environment or control.

Control events go straight to the reducer, and the state is updated.

Environment events first go to the intent classifier, which uses previous events, the current state, and the incoming event to derive the correct intent. The intent then goes into our intent executor, which converts the intent into a set of actual device changes, which gets sent to our reducer, and the state is then updated.

Lastly, we invoke the reconciliation to update our real devices to reflect our new state.

There we go! Now we have ended up with a self-contained setup. We can run it without the reconciliation or mock it to create tests for our setup and work without changing any real devices, and we can re-run the reconciliation on our state to ensure our state gets updated correctly, even if a device should miss an update.

Success!!!

But I promised to give an idea of how to take this smart home and make it actually "smart."

Let's imagine that we did not want to "program" our smart home. Instead, we wanted to use it; turning the lights on and off using the switches when we entered and exited a room, dimming the lights for movie time, and so on, and over time we want our smart home to pick up on those routines and start to do them for us.

We have a setup where we both have control events and environments coming in. Control events represent how we want the state of our home to be in a given situation. Environment events represent what happened in our home. So we could store those historically with some machine learning and look for patterns.

Let's say you always dim the light when playing a movie that is more than one hour long; your smart home would be able to recognize this pattern and automatically start to do this routine for you.

Would this work? I don't know. I am trying to get more skilled at machine learning to find out.