WTF episode 11

Managing complex UI with XState

A short introduction on State Machines, XState, and State Modeling in React.

MUHAMMAD NASRURROHMAN - HIEDU

What we’re going to talk about today

- Why you should care
- State machines, Statechart, XState
- Some XState features and example

Here’s a basic fact about software engineering:

The more states your program can be in, the more likely it is that bugs can creep in.

Example 1

What is State Machine?

A finite state machine is a mathematical model of computation that describes the behavior of a system that can be in only one state at any given time.

What Finite Machine Helps

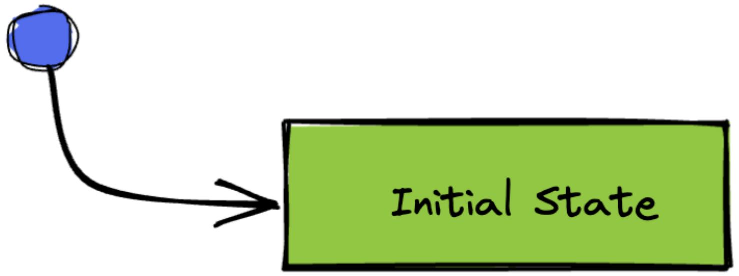
Preventing impossible states and transition

Formally, finite state machines have five parts

- An initial state
- A finite number of states
- A finite number of events
- A transition function that determines the next state given the current state and event
- A (possibly empty) set of final states

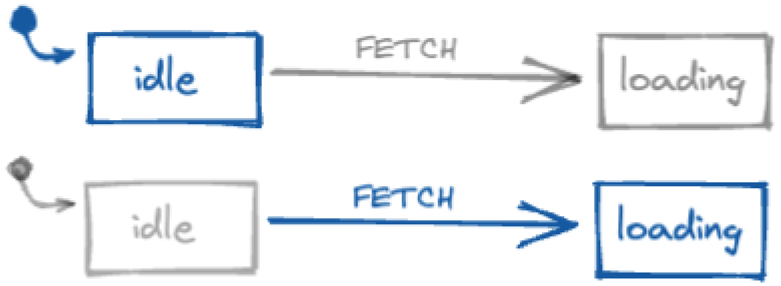
You define a finite state machine by a list of its states, its initial state and the events that trigger each transition.

An Initial State



A finite number of events

An event is a "signal" that something happened.

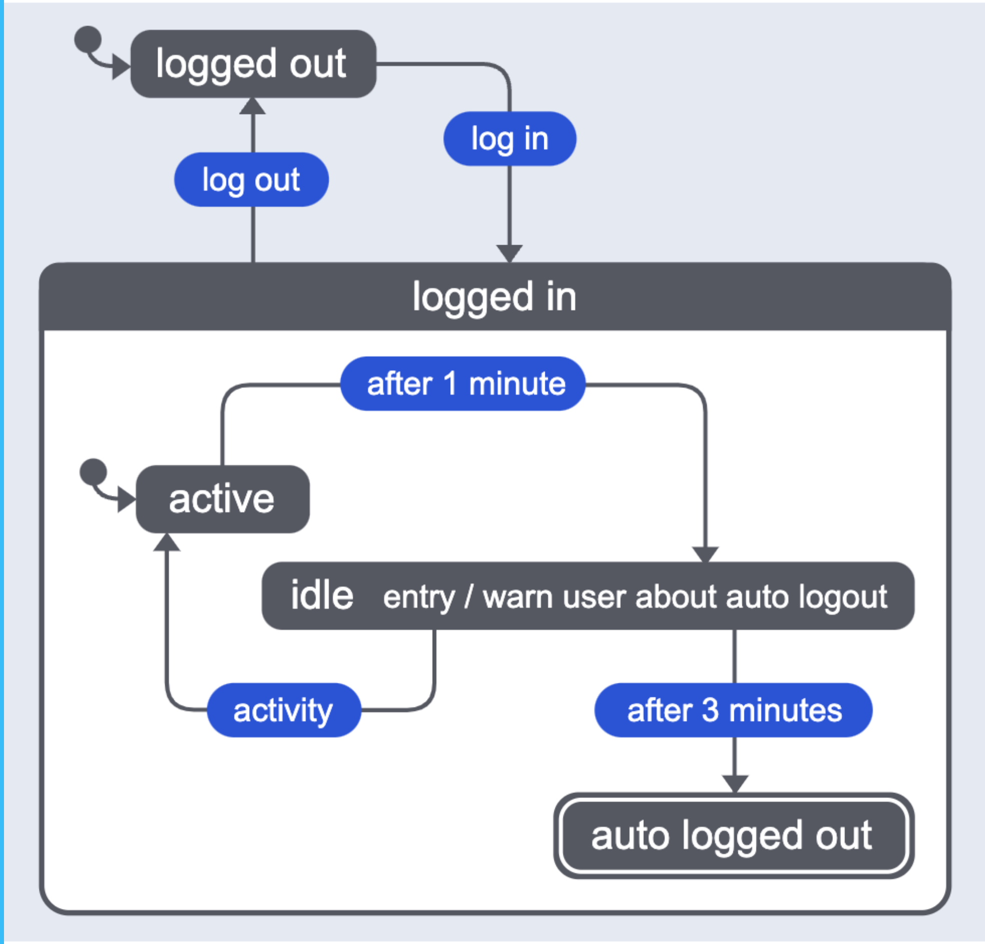


(initial state)

```
{
  type: 'FETCH',
  userId: 1
}
```

A finite number of states

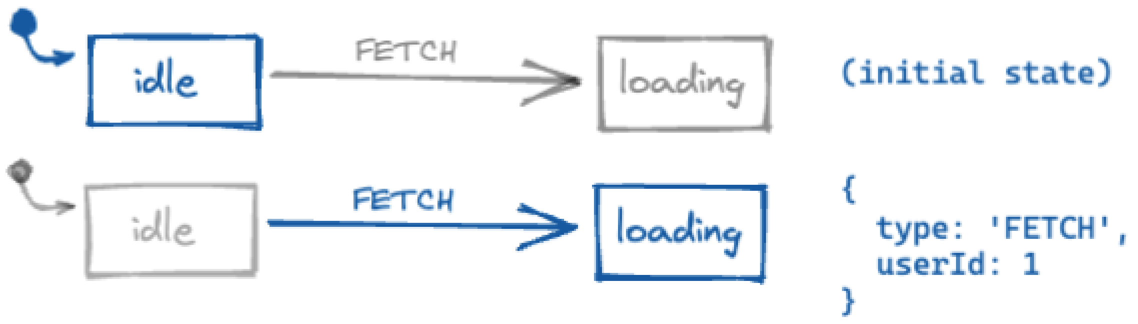
A finite state describes some "mode" or "status" that a system (e.g., an app or component) is in.



A Transition Function

Events can trigger "transitions" between states.

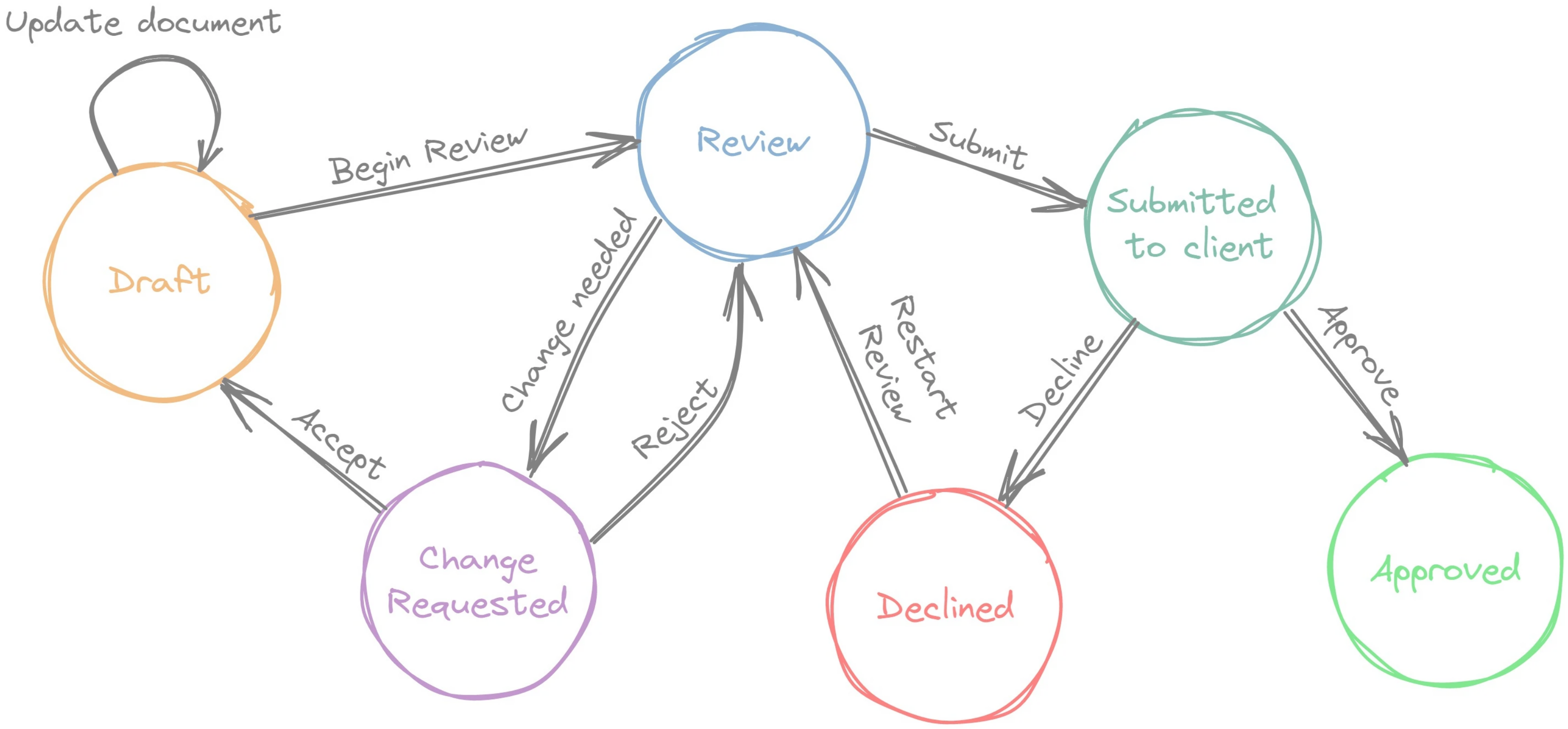
A transition is caused by an event that results in the change of state. Transitions are labelled with their events.



```
(initial state)
```

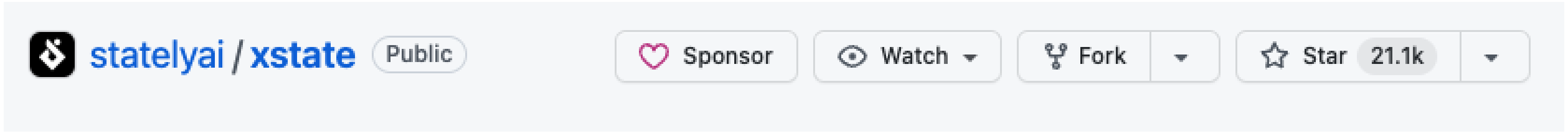
```
{
  type: 'FETCH',
  userId: 1
}
```

A (possibly empty) Set Of Final States



XState is...

a JavaScript state management library that stands upon the shoulders of giants. It embraces the W3C state chart XML spec & implements 30+ year-old formalism.



Why XState is useful?

- **Encourage planning** out your component before you create it. And planned code is better code.
- **You stand on the shoulder of giant**. State machines are a very old idea which means they've been tested and proven.
- Sometimes **diagrams communicate a lot better** than text. And Xstate visualizer is a great way of communicating what a state machine is doing

Revisit Example 1

Reimplement using XState.

XState is a state orchestrator

Redux (and similar libraries) are state containers. [Detailed diff]

Essential features / terms of XState

- Actions
- Context
- Guards
- Compound States
- Parallel states

Actions

An action is an effect that is executed as a reaction to a state transition. Actions are fire-and-forget; that is, they are executed without needing to wait for a response.

```
// ...
TRIGGER: {
  target: 'active',
  // transition actions
  actions: (context, event) => { console.log('activating...
}
// ...
```

Context

State that represents quantitative data (e.g., arbitrary strings, numbers, objects, etc.) that can be potentially infinite is represented as extended state (opens new window)instead. In XState, extended state is known as **context**.

```
import { createMachine, assign } from 'xstate';

// Action to increment the context amount
const addWater = assign({
  amount: (context, event) ⇒ context.amount + 1
});

// Guard to check if the glass is full
function glassIsFull(context, event) {
  return context.amount ⩾ 10;
}

const glassMachine = createMachine(
  {
    id: 'glass',
    // the initial context (extended state) of the statecha
    context: {
      amount: 0
    },
    initial: 'empty',
    states: {
      empty: {
        on: {
          // the convention is to use CONST_CASE for event
          FILL: {
            target: 'filling',
            actions: 'addWater'
```

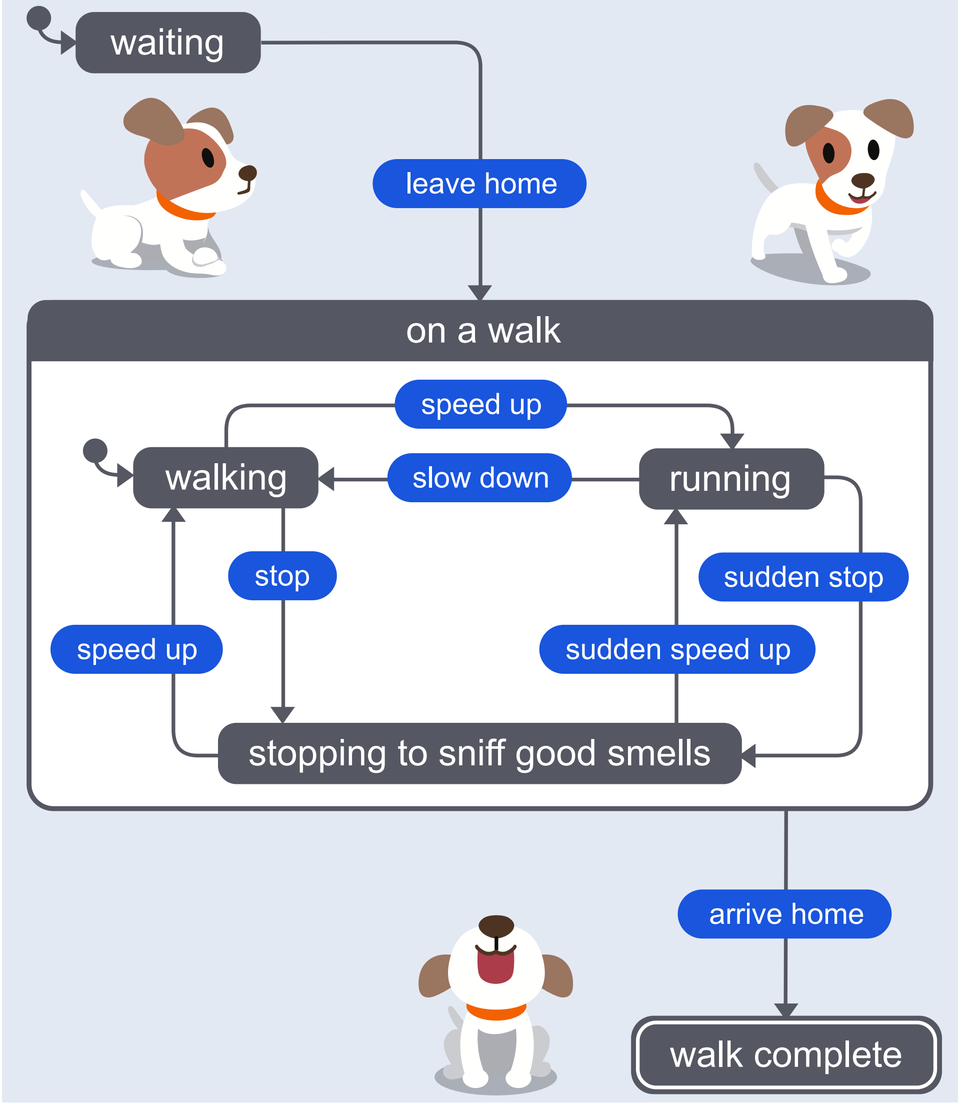
Guards

A transition with condition(s) is called a **guarded transition**.

```
///
idle: {
  on: {
    SEARCH: [
      {
        target: 'searching',
        // Only transition to 'searching' if the guard (cond) evaluates to true
        cond: searchValid // or { type: 'searchValid' }
      },
      { target: '.invalid' }
    ]
  },
///
```

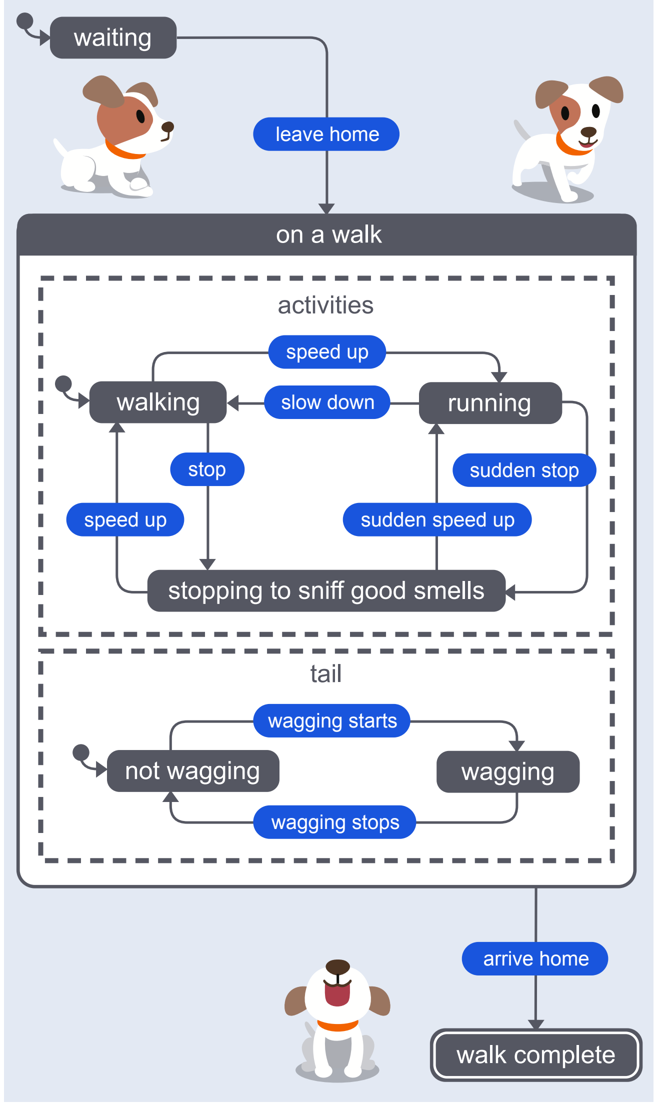
Compound States

A compound state has one or more child states.



Parallel states

A parallel state is a compound state where all of its child states, also known as regions, are active simultaneously.



Example

Utilize `` `invoke` `` property + `` `guards` ``

Resources

- https://xstate.js.org/
- https://xstate.js.org/docs/guides/
- https://github.com/darrylhebbes/awesome_xstate

Thank You

email: anas@jurnalanas.com