
UI modeling with statecharts
How to make sense of complex UI logic?

Motivation (1/2) - Facetime bug (January 2019)

1. Start a FaceTime video call
2. Before the call is answered, tap “Add Person” and add yourself
3. You can listen to the callee’s microphone before they accept the call

The app was in a state that should be impossible.

https://medium.com/@DavidKPiano/the-facetime-bug-and-the-dangers-of-implicit-state-machines-a5f0f61bdaa2

Motivation (2/2) - Dangers of implicit state machines

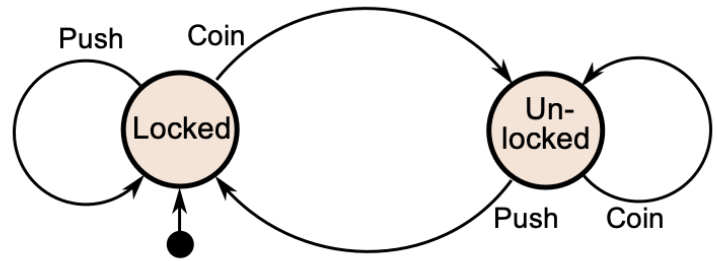
State machines are already hiding in your code :-)

```
const [loading, setLoading] = useState(false);
const [uploadError, setUploadError] = useState(false);

const onDrop = async files => {
  setLoading(true);
  setUploadError(false);
  try {
    const values = await upload(files);
    setLoading(false);
    onChange(values);
  } catch (e) {
    setLoading(false);
    setUploadError(true);
    setTimeout(() => {
      setUploadError(false);
    }, 2000);
  }
};
```

State machines (1/2)

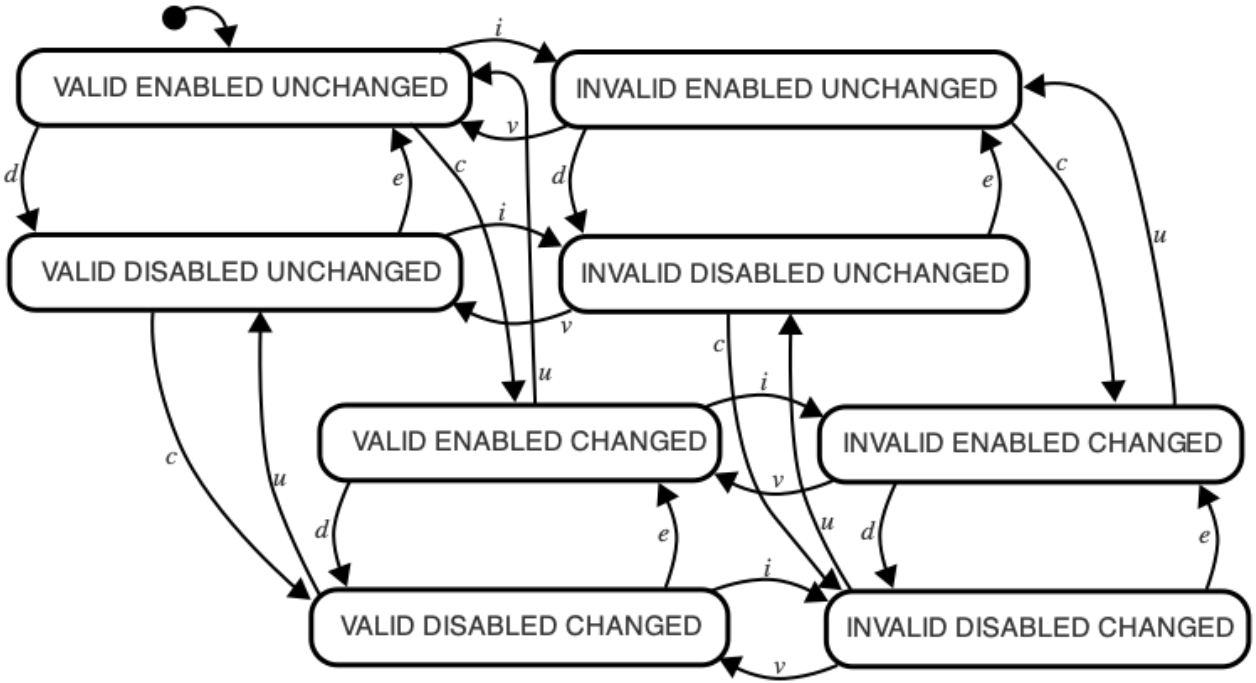
- A finite number of states
- A finite number of events
- An initial state
- A transition function that determines the next state given the current state and event
- A (possibly empty) set of final states



A state machine diagram for a turnstile

https://en.wikipedia.org/wiki/Finite-state_machine#/media/File:Turnstile_state_machine_colored.svg

State machines (2/2) - Problem: state explosion



https://statecharts.dev/valid-invalid-enabled-disabled-changed-unchanged.svg

Statecharts

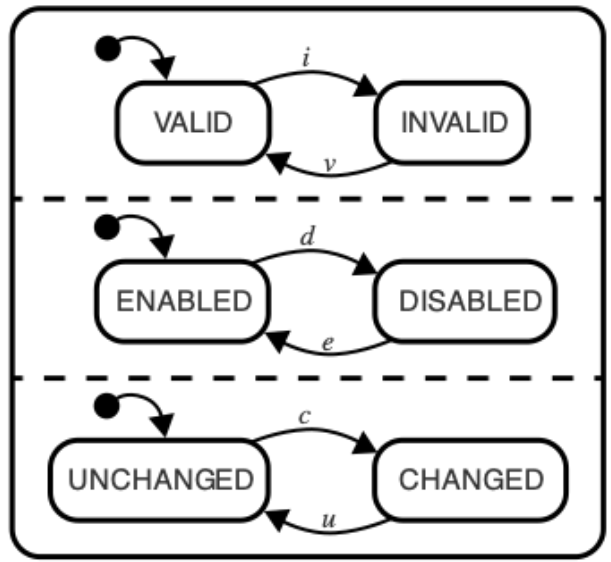
"A visual formalism for complex systems"

Extended state machines. Some of the extensions include:

- Guarded transitions
- Actions (entry, exit, transition)
- Extended state (context)
- Orthogonal (parallel) states
- Hierarchical (nested) states
- History

https://www.sciencedirect.com/science/article/pii/0167642387900359/pdf

https://statecharts.dev/valid-invalid-enabled-disabled-changed-unchanged-parallel.svg



XState (1/4) - Executable statecharts

- A Framework-agnostic statechart library
- Has bindings for React, Vue and Svelte
- Developed by a startup called Stately

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

const promiseMachine = createMachine({
  id: 'promise',
  initial: 'pending',
  states: {
    pending: {
      on: {
        RESOLVE: { target: 'resolved' },
        REJECT: { target: 'rejected' }
      }
    },
    resolved: {
      type: 'final'
    },
    rejected: {
      type: 'final'
    }
  }
});
```

XState (2/4) - Context, actions and guards

```
const states = {
  empty: {
    on: {
      FILL: {
        target: 'filling',
        actions: 'addWater'
      }
    }
  },
  filling: {
    // Transient transition
    always: {
      target: 'full',
      cond: 'glassIsFull'
    },
    on: {
      FILL: {
        target: 'filling',
        actions: 'addWater'
      }
    }
  },
  full: {}
}
```

```
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
const glassIsFull = function (context, event) {
  return context.amount >= 10;
};

const glassMachine = createMachine({
  id: 'glass',
  // Extended state
  context: {
    amount: 0
  },
  initial: 'empty',
  states,
},
{
  actions: { addWater },
  guards: { glassIsFull }
});
```

XState (3/4) - Invoking a promise

```
const fetchUser = (userId) =>
  fetch(`url/to/user/${userId}`)
    .then((response) => response.json());

const loading = {
  invoke: {
    id: 'getUser',
    src: (context, event) =>
      fetchUser(context.userId),
    onDone: {
      target: 'success',
      actions: assign({ user: (context, event) => event.data })
    },
    onError: {
      target: 'failure',
      actions: assign({ error: (context, event) => event.data })
    }
  }
}
```

```
const userMachine = createMachine({
  id: 'user',
  initial: 'idle',
  context: {
    userId: 42,
    user: undefined,
    error: undefined
  },
  states: {
    idle: {
      on: {
        FETCH: { target: 'loading' }
      }
    },
    loading,
    success: {},
    failure: {
      on: {
        RETRY: { target: 'loading' }
      }
    }
  }
});
```

XState (4/4) - Actors

```
import { createMachine, spawn } from 'xstate';
import { todoMachine } from './todoMachine';

const todosMachine = createMachine({
  // ...
  on: {
    'NEW_TODO.ADD': {
      actions: assign({
        todos: (context, event) => [
          ...context.todos,
          {
            todo: event.todo,
            // add a new todoMachine actor with a unique name
            ref: spawn(todoMachine, `todo-${event.id}`)
          }
        ]
      })
    }
    // ...
  }
});
```

Workshop

- Implement UI logic for a wordle-clone (https://wordlegame.org/)
- Repo here: https://github.com/KnowitJSTSGuild/ui-modeling-with-statecharts
- Workshop template is in the **exercise**-directory
- UI components and some other utilites are there, you just need to implement the logic with xstate
- You can use the Stately editor, if you want (https://stately.ai/registry/new)
- This is not an exam. If you get stuck, peek the **solution**-directory or ask for help in the chat