Animating Components with redux-time

By Juan Diego Garcia
@juandgoc
linkedin.com/in/juandgoc

Background

- Python / Django
- React / Redux
- Full Stack Dev / Oddslingers

What Is an Animation?

- A start_state and end_state
- A start_time and end_time
- A curve_function

What Is an Animation?

```
<div style="top: 100px" id="demo">abc</div>

$('demo').animate({top: 200}, 2000)
setTimeout(function() {
    $('demo').animate({top: 100}, 2000)
}, 2000)
```

Patterns

- Imperative stateful animation
- Pure functional animation
- Declarative animation

Imperative Animation

```
start_position = 15
step_size = 3

animate() {
    current_position = start_position   // state that is updated on each run
    while true {
        current_position += step_size
        render(current_position)
        sleep(14ms)
    }
}
```

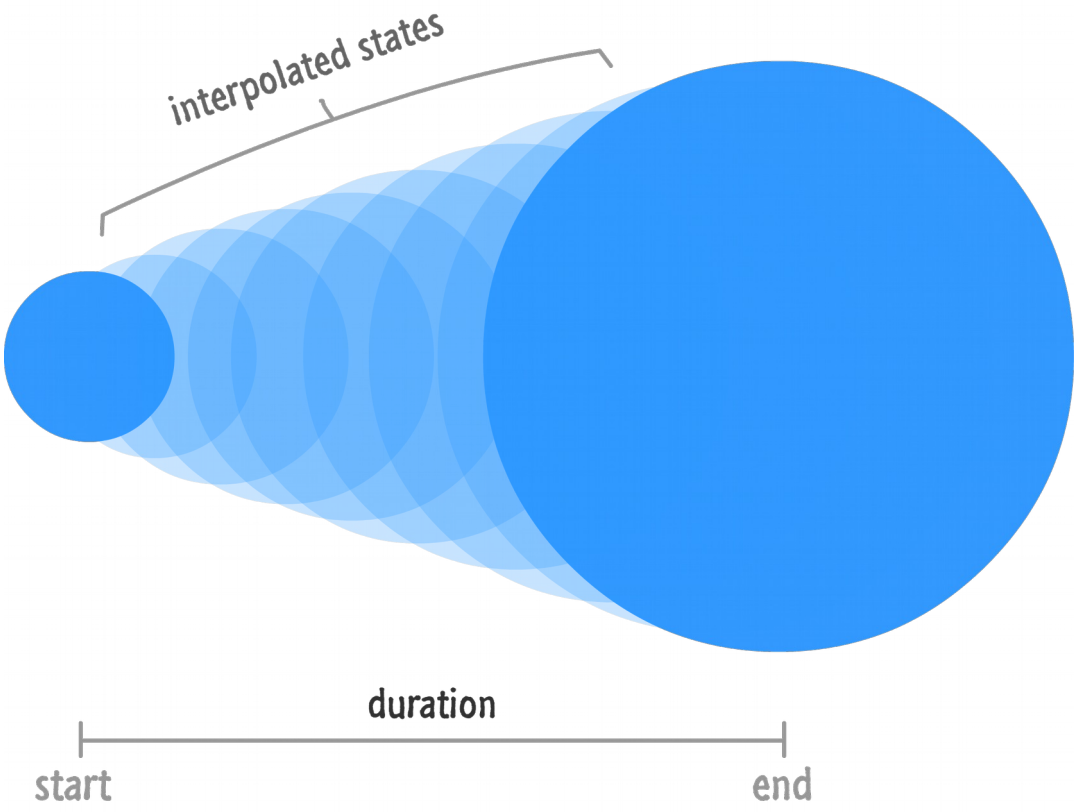
Functional Animation

```
start_position = 15
step_size = 3

ballPosition(step) {
    return start_position + step * step_size
}

animate() {
    for (step_number=0; true; step_number++) {
        render(ballPosition(step_number))
        sleep(14ms)
    }
}
```

Declarative Animation

```
const rotate = {
    path: '/button/style/transform/rotate',
    start_time: 1540702920000,
    duration: 1000,
    start_state: 0,
    end_state: 180,
    unit: 'deg',
    curve: 'easeInOutQuad',
}
```

Declarative Animation



Implementing a Render Loop

First we define the animation as data.

```
// grow from 0% width to 100%
const grow = {
    start_time: 1540702921000,
    duration: 2000,
    start_state: 0,
    end_state: 100,
    unit: '%',
}
```

Implementing a Render Loop

Then we render this animation in a loop.

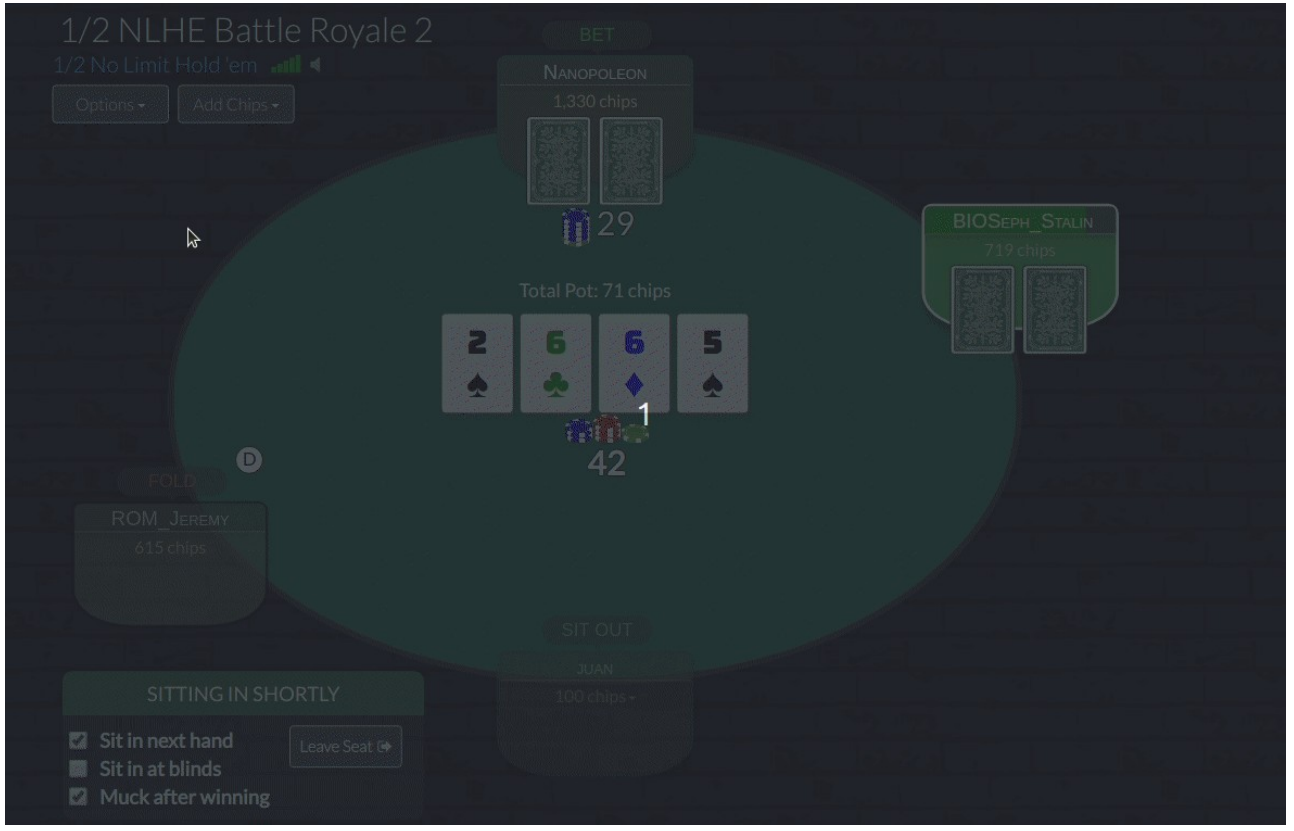
```
const render = (timestamp) => {
    const width = compute_animated_state(grow, timestamp)
    document.getElementById('demo').style.width = width
    window.requestAnimationFrame(render)
}
```

Implementing a Render Loop

We compute animated state based on the current timestamp and curve function.

```
// given animation description and current timestamp
const compute_animated_state = (anim, ts) => {
    const start_time = anim.start_time
    const duration = anim.duration
    const end_time = start_time + duration

    // if animating, calculate intermediate animated value
    if (ts >= start_time && ts <= end_time) {
        const delta_total = anim.end_state - anim.start_state
        const progress = (ts - start_time) / duration
        return start_state + (progress * delta_total)
    }

    // if not currently animating, return start or end state
    return ts > end_time
        ? end_state
        : start_state
}
```

Redux-time


1/2 NLHE Battle Royale 2
1/2 No Limit Hold 'em
Options
Add Chips
BET
NANOPOLEON
1,330 chips
29
BIOSEPH_STALIN
719 chips
Total Pot: 71 chips
2
6
6
5
1
42
FOLD
ROM_JEREMY
615 chips
SIT OUT
JUAN
100 chips
SITTING IN SHORTLY
Sit in next hand
Sit in at blinds
Muck after winning
Leave Seat

Redux-time

- Compute your state tree as a function of time
- Generically changing any redux state as time progresses
- React/React-Native
- ThreeJS
- Canvas

Walkthrough Example

```
import {animationsReducer, startAnimation} from 'redux-time'
import {Animate} from 'redux-time/node/animations'

// 1. Create a redux store, and start the animation runloop with initial state
const store = createStore(combineReducers({animations: animationsReducer}))

const initial_state = {ball: {style: {}}}
const time = startAnimation(store, initial_state)
```

Walkthrough Example

```
// 2. Set up our first animation
const move_ball_animation = Animate({
    // move the ball 20px down over 1s
    path: '/ball/style/top',
    start_state: 0,
    end_state: 20,
    duration: 1000,
})

document.onkeypress = (e) => {
    // trigger it when the down arrow is pressed
    if (e.keyCode == 40) {
        store.dispatch({type: 'ANIMATE', animation: move_ball_animation})
    }
}
```

Walkthrough Example

```
// 2. Create a component to display our state
const BallComponent = ({ball}) =>
    <div style={{position: 'absolute', ...ball.style}}></div>

const mapStateToProps = ({animations}) => ({
    ball: animations.state.ball,
    // optionally deepMerge(other_state, animations.state)
})
const Ball = connect(mapStateToProps)(BallComponent)
```

Walkthrough Example

```
// 4. Then render it
ReactDOM.render(
    <Provider store={store}>
        <Ball/>
    </Provider>,
    document.getElementById('react')
)
```

Live Example

Live Example !!!

Thanks

Thank you