Finite state machine...

...and some retrogaming
It is an abstract machine that can be in exactly one of a finite states at any given time.

It can change from a state to another in response to some inputs.

A FSM can be represented by a connected graph, called statechart where the nodes are the states, and the links are the transitions.
Example /elevator
Let’s create a statechart
Create statechart / states

Idle
Create statechart / states

Idle

Boxer
Create statechart / states

- Idle
- Boxer
- Military
Create statechart / transitions

Idle → Marco is near → Boxer

Military
Create statechart / transitions

Idle → Marco is near → Boxer

Military → Reward dropped
Create statechart / transitions

- Idle
- Marco is near
- Boxer
- Reward dropped
- Marco is far
- Military

source sense: Open Solutions for your Value
Simplest FSM code

```javascript
// javascript
class Fsm {
  setState = (state) => {
    this.activeState = state; // activeState must be a function!
  }

  update = () => {
    if(this.activeState) {
      this.activeState();
    }
  }

  export default Fsm;
}
```
Edge case

credits: https://gamedevelopment.tutsplus.com/
Edge case

credits: https://gamedevelopment.tutsplus.com/
Stack based FSM

- Stack of states instead of active state
- Active state is the one on top of the stack
- Every state must pop itself from the stack at the right time
Stack based FSM

```javascript
1 // javascript
2
3 class FsmStack {
4   constructor() {
5     this.stack = [];
6   }
7
8   popState = () => this.stack.pop();
9
10  pushState = (state) => this.stack.push(state);
11
12  currentState = () => this.stack[0];
13
14  update = () => {
15    const active = this.currentState();
16    if (active) {
17      active();
18    }
19  }
20 }
```
Javascript and Typescript finite state machines and statecharts for modern web
import { createMachine, interpret } from 'xstate';

// Stateless machine definition
// machine.transition(...) is a pure function used by the interpreter.
const toggleMachine = createMachine({
  id: 'toggle',
  initial: 'inactive',
  states: {
    inactive: { on: { TOGGLE: 'active' } },
    active: { on: { TOGGLE: 'inactive' } }
  }
});

// Machine instance with internal state
const toggleService = interpret(toggleMachine)
  .onTransition(state => console.log(state.value))
  .start();

// => 'inactive'
toggleService.send('TOGGLE');
// => 'active'
toggleService.send('TOGGLE');
// => 'inactive'
Code to visual statechart visualizer. (let’s see on the site)
Questions?

Gabriele Falasca
Frontend dev @Sourcesense
Mozilla Tech Speaker
@gabrycaos