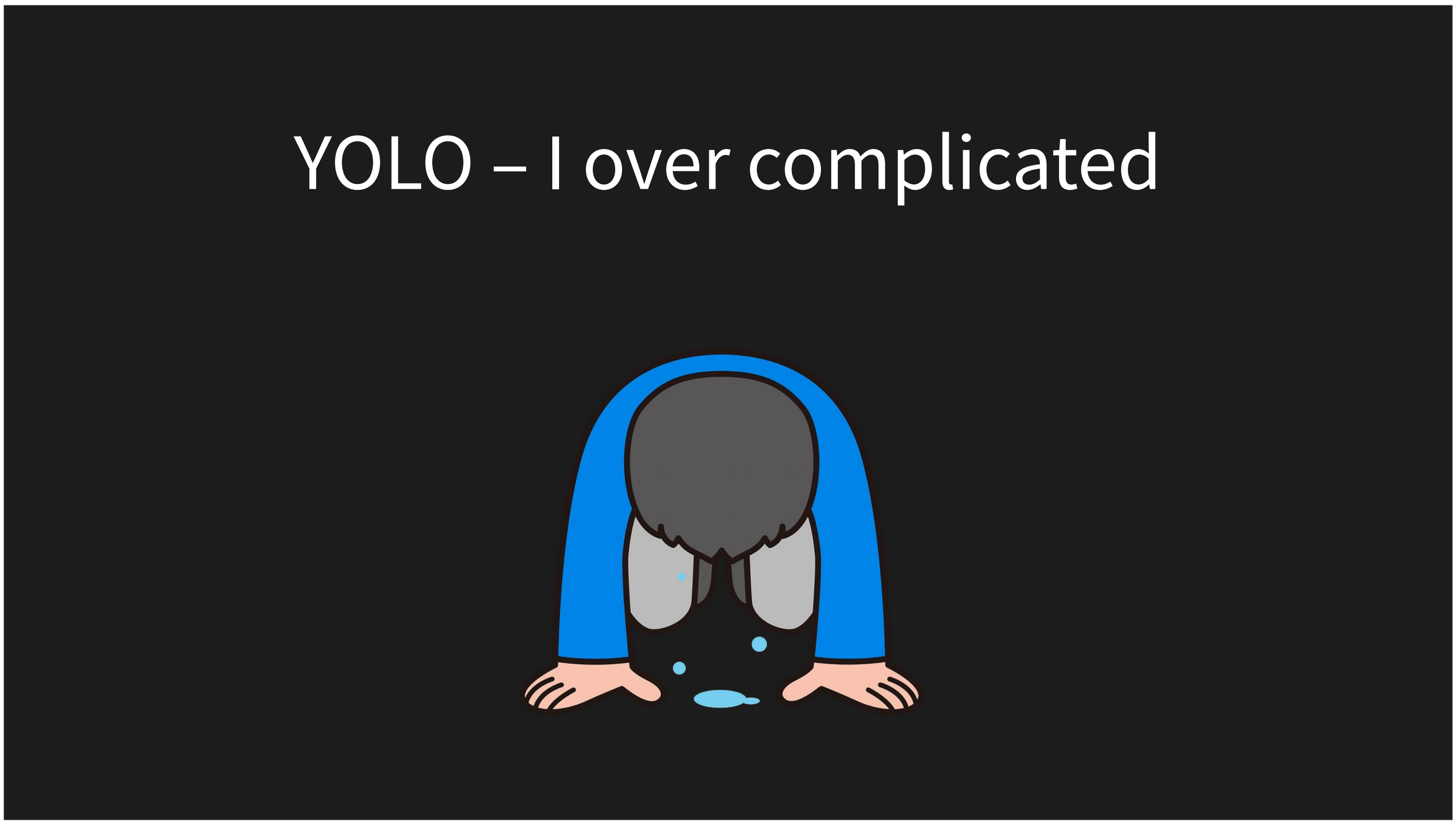
Contemporary Web Development Lesson 12

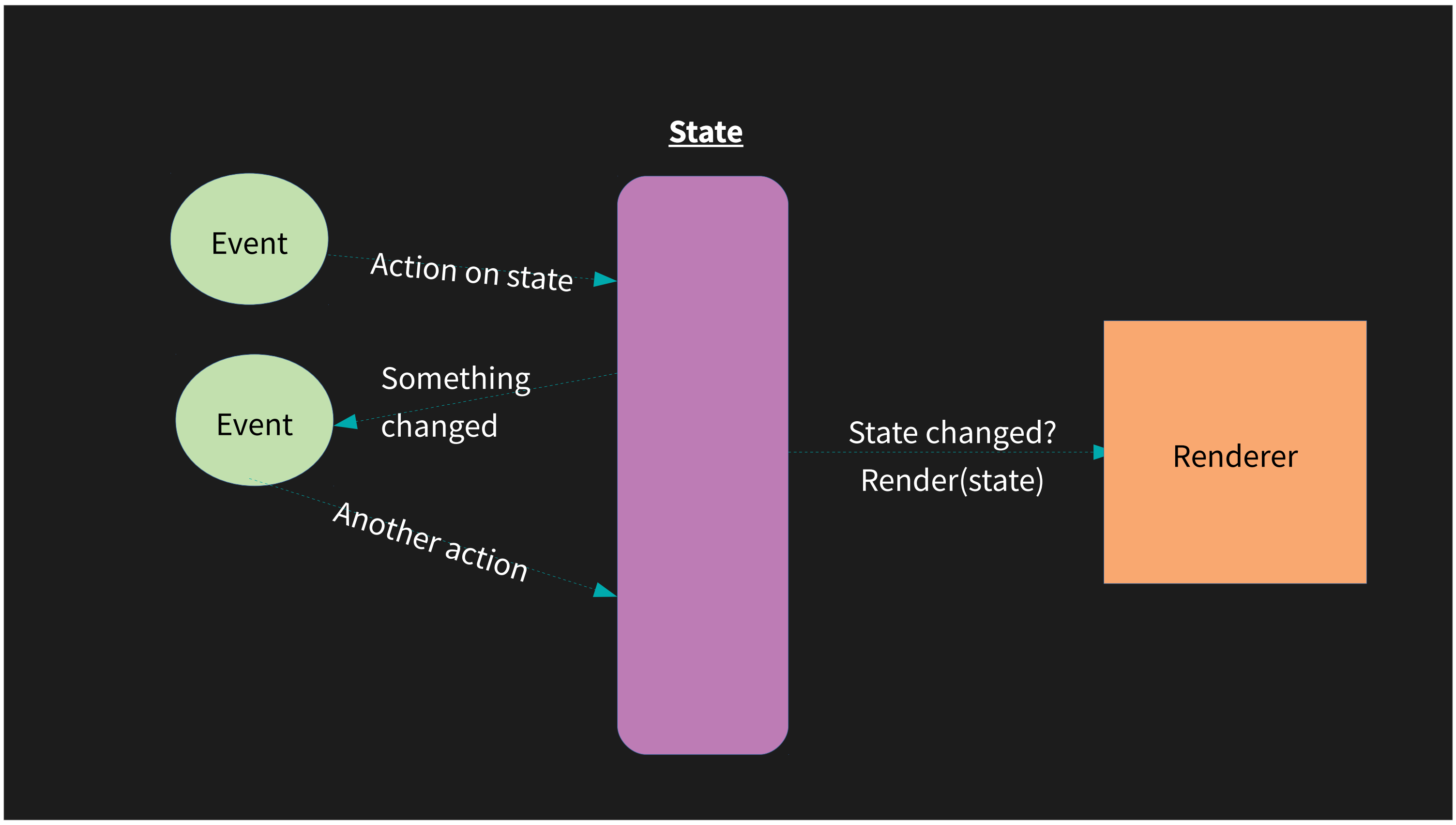
https://www.youtube.com/playlist?list=PLtP6Zh5THBgVvyhcPlbZheeKq3vMJUDBN



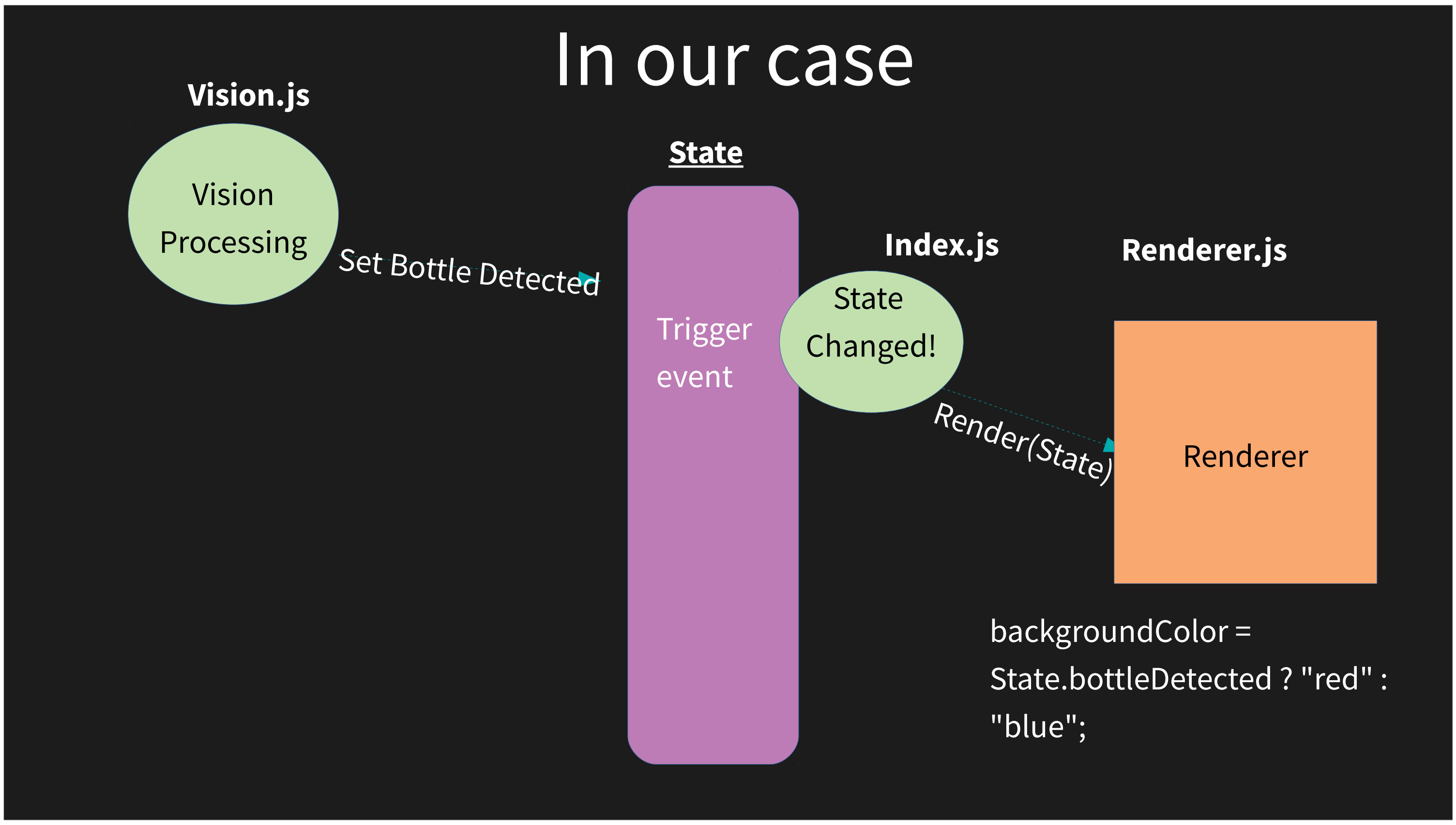
YOLO – I over complicated

The actual HTML rendering should make the choice based on the state.

It may very well happen that a state change triggers another state change, which is what I tried to simulate with that example. But it shouldn’t happen on the visual elements,


State
Event
Action on state
Event
Something changed
Another action
State changed?
Render(state)
Renderer

In our case



```
backgroundColor =
State.bottleDetected ? "red" :
"blue";
```

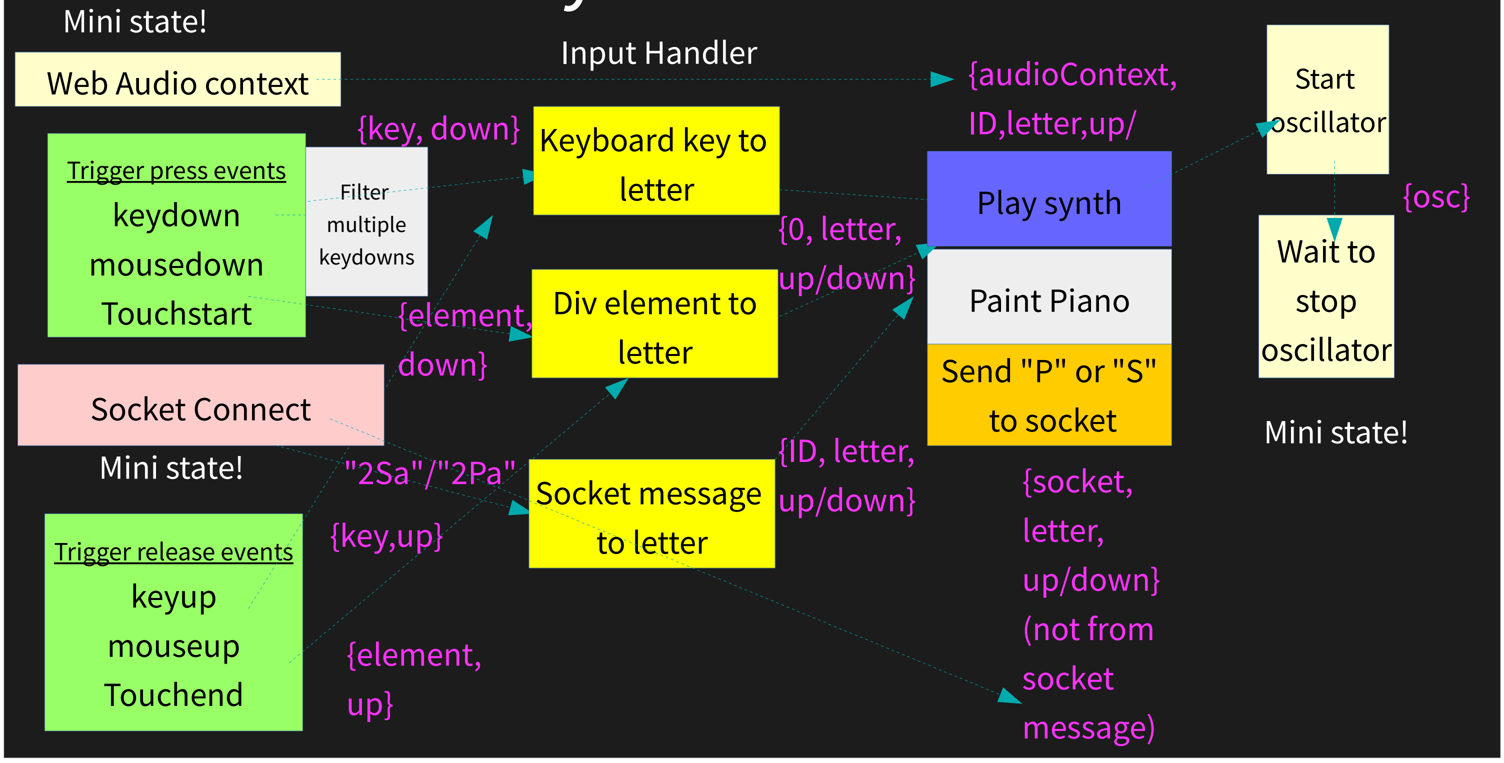
state.js

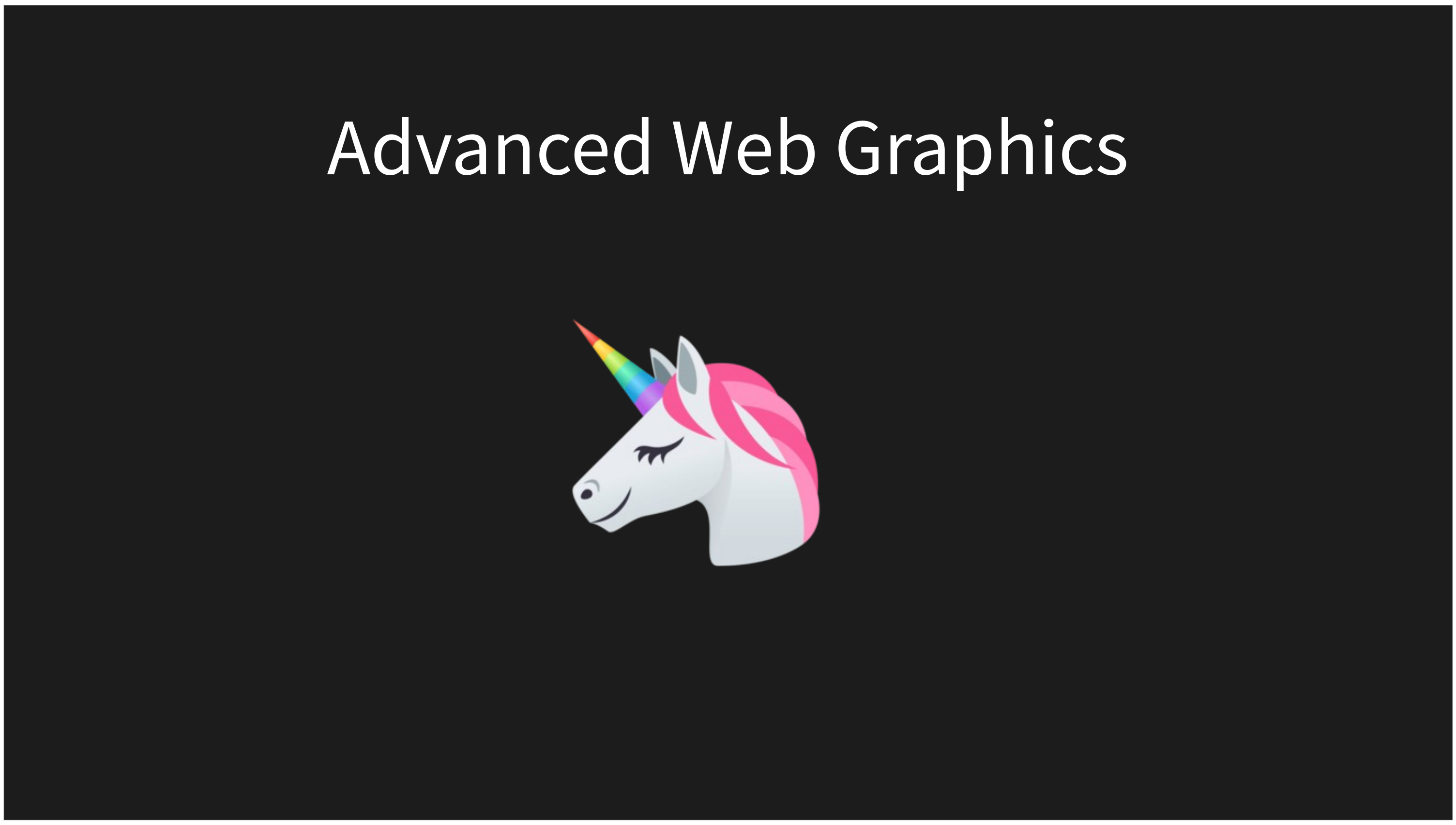
```
export const Vision = {
   bottlePresent: false,
   setBottlePresent: function(present) {
        // Trigger if state was changed
    }
};
observable(Vision);

export const Render = {};
```

Index.js

```
 State.Vision.on('bottle-present-updated',
(bottlePresent) => {
       Renderer.drawBackground(State,
elements.boxesElement);
   })
```

Collab Synth - Stream Flow





Advanced Web Graphics

The animation loop

```
function step(timestamp) {
        gameLoop.run();
        window.requestAnimationFrame(step);
}

window.requestAnimationFrame(step);
```

2D

PixiJS v4

3D

three.js

Game Engines / Platforms

p5*

unity WebGL

PHASER

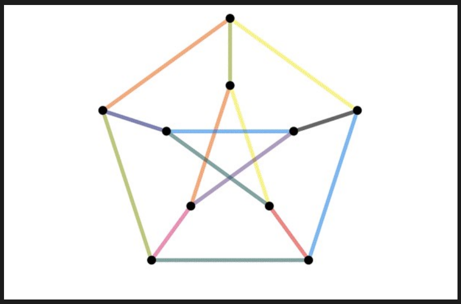
CONSTRUCT3

Web Assembly / Emscripten

Quake 3 JS

WebXR

Isomorphic Web Applications



The search engine problem

Google’s search console

The observable state approach

My own attempt: Riot-Isomorphic

A common router

Using Page.js and Page-Express-Mapper

Route → Actions → Rendering

In the sever the state has to be ‘populated’ before rendering starts, while in the client everything renders as soon as the data is updated;

Merging observables

<u>Trigger press events</u>

keydown

mousedown

Touchstart

```
let keys = document.querySelector("#keys");
const mouseDown$ = fromEvent(keys, "mousedown").pipe(
   map(event => event.target.id)
);
const mouseUp$ = fromEvent(keys, "mouseup").pipe(
   map(event => event.target.id)
);
const keyDown$ = fromEvent(document, "keydown").pipe(
         map(event => keyEventToLetter(event))
);
const keyUp$  = fromEvent(document, "keyup").pipe(
   map(event => keyEventToLetter(event))
);
const touchStart$ = fromEvent(keys, "touchstart").pipe(
   map(event => event.target.id)
);
const touchEnd$ = fromEvent(keys, "touchend").pipe(
   map(event => event.target.id)
);
```

```
const inputDown$ = merge(
        mouseDown$,
        ketDown$,
        touchStart$
);

const inputUp$ = merge(
        mouseUp$,
        ketUp$,
        touchEnd$
);
```

Initializing AudioContext

Web Audio context

```
const audio$ = inputDown$.pipe(
   first(),
   map((event) => {
      console.log("Creatign new audio
context!", event);
      return new (window.AudioContext ||
window.webkitAudioContext)()
   }),
   multicast(new Subject())
)
audio$.connect();
```

```
   observable.pipe(
      …,
      …,
      withLatestFrom(audio$),

).subscribe((audio) => {
      // Do something with audio
      // Would only work after at least
one inputDown$
});
```