Jesus Ortega

Async Notes Part 5

Sequencing Animations

We want to update a page to apply an animation. If we have three images we want to apply the animation one after the other. The animation is defined below in main.js file:

```
const aliceTumbling = [
  { transform: 'rotate(0) scale(1)' },
  { transform: 'rotate(360deg) scale(0)' }
];

const aliceTiming = {
  duration: 2000,
  iterations: 1,
  fill: 'forwards'
}

const alice1 = document.querySelector("#alice1");
const alice2 = document.querySelector("#alice2");
const alice3 = document.querySelector("#alice3");
```

The code above rotates the image and shrinks it until it disappears.

Animating The First Image

We're using the **Web Animations API** to animate the images, specifically the **element.animate()** method.

Wer're gonna update the main.js code to call alice1.animate():

```
const aliceTumbling = [
  { transform: "rotate(0) scale(1)" },
  { transform: "rotate(360deg) scale(0)" },
];

const aliceTiming = {
  duration: 2000,
  iterations: 1,
  fill: "forwards",
```

```
};

const alice1 = document.querySelector("#alice1");
const alice2 = document.querySelector("#alice2");
const alice3 = document.querySelector("#alice3");

alice1.animate(aliceTumbling, aliceTiming);
```

Now the first image will rotate and shrink.

Animating All The Images

The **animate()** method returns an Animation Object. This object has a finished property, which is a **Promise** that is fulfilled when the animation has finished playing. We can use this Promise to know when to start the next animation.

Different Ways of Implementing Promises

1. First, implement something that works, but has the promise version of the "callback hell" problem.
2. Next, implement it as a promise chain.
3. Finally, implement it using async and await.

NOTE: **element.animate()** does not return a **Promise**: it returns an Animation object with a *finished* property that is a **Promise**.