Animate elements in a sequence

Posted on January 24, 2022 | by Prashant Yadav

Posted in Interview, Javascript | Tagged SystemDesign

I have found this question on leetcode where it was asked to animate elements in a sequence.

The problem statement is,

- Implement a loading bar that animates from 0 to 100% in 3 seconds.
- Start loading bar animation upon a button click.
- Queue multiple loading bars if the button is clicked more than once. Loading bar N starts animating with loading bar N-1 is done animating.

We will implement it in vanilla JavaScript as well as in React along with its two variations.

1. Animated loading bars in batches.
2. Start loading `N` bar after `N-1` is half done (50%).

As the problem statement can be conquered individually let's start solving it.

A loading bar that animates

Create a div dynamically through JavaScript.

Practically prepare for your JavaScript interview

JavaScript Revision
JavaScript- Concept Based Problems
Data Structures
Algorithms
Machine Coding
Web Fundamentals

```
const loadingBar = document.createElement("div");
```

Apply the styles.

The most challenging part which I had found for this was applying dynamic animation keyframes to the element, after a quick google search I found the solution.

```
let styleSheet = null;
const dynamicAnimation = (name, styles) => {
    //create a stylesheet
    if (!styleSheet) {
        styleSheet = document.createElement("style");
        styleSheet.type = "text/css";
        document.head.appendChild(styleSheet);
    }

    //insert the new key frames
    styleSheet.sheet.insertRule(
        `@keyframes ${name} {${styles}}`,
        styleSheet.length
    );
};
```

Using this function we can dynamically add the keyframes of the animation and then apply these animations to any element.

```
dynamicAnimation(
    "loadingBar",
    `
    0%{
        width: 0%;
    }
    100%{
        width: 100%;
    }`
);
loadingBar.style.height = "10px";
loadingBar.style.backgroundColor = "Red";
loadingBar.style.width = "0";
loadingBar.style.animation = "loadingBar 3s forwards";
```

We are done creating the loading bar, just need to add it to the DOM to animate, for which we will get an entry element and append this into that.

```
const entry = document.getElementById("entry");
entry.appendChild(loadingBar);
```

Wrap everything inside a function and invoke it to generate a loading bar. You can also pass the duration to this function (how long the animation should run) as well as the keyframes iteself.

```
const generateLoadingBar = () => {
    //create a div
    const loadingBar = document.createElement("div");

    //apply styles
    dynamicAnimation(
        "loadingBar",
        `
        0%{
            width: 0%;
        }
        100%{
            width: 100%;
        }`
    );
    loadingBar.style.height = "10px";
    loadingBar.style.backgroundColor = "Red";
    loadingBar.style.width = "0";
    loadingBar.style.marginBottom = "10px";
    loadingBar.style.animation = "loadingBar 3s forwards";

    //append the div
    const entry = document.getElementById("entry");
    entry.appendChild(loadingBar);
};
```

Start loading bar animation upon a button click.

Create a button and on its click invoke the above function so that it will generate the loading bar and will be animated once added to the DOM.

```
//on btn click, generate the loading bar
document.getElementById("btn").addEventListener("click", (e) => {
   generateLoadingBar();
});
```

Queue multiple loading bars if the button is clicked more than once and load the next one once the previous animation is finished.

The last part of this question is to queue the loading bars when the button is clicked multiple times and animate them sequentially one after another.

Create a global variable `count` and increment its value by one every time the button is clicked, vice-versa decrease its value by one, when a loading bar is animated.

Copy

```
//global variable to track the count of loading bars
let count = 0;

//function to update the count
const updateCount = (val) => {
    count += val;
    document.getElementById("queueCount").innerText = count;
};
```

For generating the next loading bar from the queue, we will have to recursive call the same function (which generates the loading bar) when the animation of the previous loading bar is done.

Thankfully we have an event for that `animationend` which is fired every time an animation ends on the element, this is also a reason why I choose CSS animation over JavaScript timers to animate elements.

When the `animationend` is triggered, recursively call the same function to generate the loading bar and update the queue count.

Copy

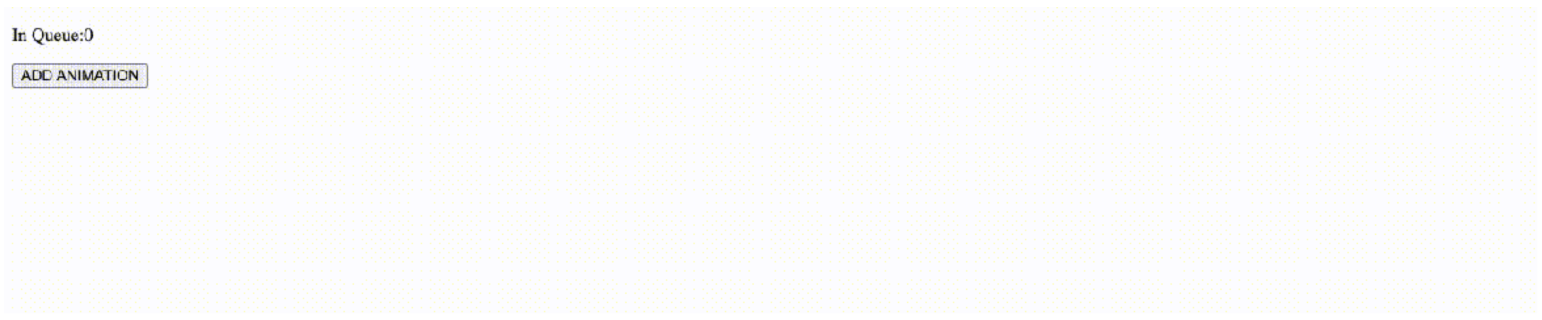
```
//on animation end
loadingBar.addEventListener("animationend", () => {
    //decrease the count
    updateCount(-1);

    if (count > 0) {
      //generate the loading bar
      generateLoadingBar();
   }
});
```

We will also need to update the code on the button click, invoke the `generateLoadingBar` function only when the `count` is zero as all other subsequent calls will be invoked recursively.

Also, update the count on each click.

Copy

```
//on btn click, generate the loading bar
document.getElementById("btn").addEventListener("click", (e) => {
    //trigger animation
    if (count === 0) {
        generateLoadingBar();
    }

    //update count
    updateCount(1);
});
```



Complete code

Copy

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>Animate elements in sequence</title>
  </head>
  <body>
    <div id="entry"></div>
    <p><span>In Queue:</span><span id="queueCount">0</span></p>
    <button id="btn">ADD ANIMATION</button>
    <script>
      // function to add keyframes dynamically
      let styleSheet = null;
      const dynamicAnimation = (name, styles) => {
        //create a stylesheet
        if (!styleSheet) {
          styleSheet = document.createElement("style");
          styleSheet.type = "text/css";
          document.head.appendChild(styleSheet);
        }

        //insert the new key frames
        styleSheet.sheet.insertRule(
          `@keyframes ${name} {${styles}}`,
          styleSheet.length
        );
      };

      //global variable to track the count of loading bars
      let count = 0;

      //function to update the count
      const updateCount = (val) => {
        count += val;
        document.getElementById("queueCount").innerText = count;
      };

      //generate loading bars
      const generateLoadingBar = () => {
        //create a div elm
        const loadingBar = document.createElement("div");

        //apply styles
        //animation keyframes
        dynamicAnimation(
          "loadingBar",
          `
        0%{
            width: 0%;
        }
        100%{
            width: 100%;
        }`
        );
        loadingBar.style.height = "10px";
        loadingBar.style.backgroundColor = "Red";
        loadingBar.style.width = "0";
        loadingBar.style.marginBottom = "10px";
        loadingBar.style.animation = "loadingBar 3s forwards";

        //append the loading bar
        const entry = document.getElementById("entry");
        entry.appendChild(loadingBar);

        //on animation end
        loadingBar.addEventListener("animationend", () => {
           //decrease the count
            updateCount(-1);

          if (count > 0) {
            //generate the loading bar
            generateLoadingBar();
          }
        });

        //remove listener
        loadingBar.removeEventListener("animationend", () => {});
      };

      //on btn click, generate the loading bar
      document.getElementById("btn").addEventListener("click", (e) => {
        //trigger animation
        if (count === 0) {
          generateLoadingBar();
        }

        //update count
        updateCount(1);
      });
    </script>
  </body>
</html>
```

Follow-up:- Loading bar N starts animating with loading bar N-1 is done animating 50%.

In the follow-up, we have to start animating the Nth bar when the N-1th bar is half done.

Unfortunately, there are only four events associated with animations.

- `animationstart`:- When animation starts.
- `animationend`:- When animation ends.
- `animationcancel`:- When animation unexpectedly aborts without triggering `animationend` event.
- `animationiteration`:- When an iteration of animation ends and next one begins. This event is not triggered at the same time as the `animationend` event.

There is no way to determine how much animation has been completed.

To solve this problem, we use a hack, a workaround, we animate two elements simultaneously, one which runs on normal duration and the other which runs for the duration when the next animation has to be triggered.

For example, the next animation should trigger when the first loading bar is 50% done, thus let’s say our original loading bar is going to complete 100% animation in 3 seconds, which means the next animation should be triggered when it is 50% done in 1.5 seconds (half time).

We will parallelly animate another element for that duration and on its `animationend` trigger the next rendering.

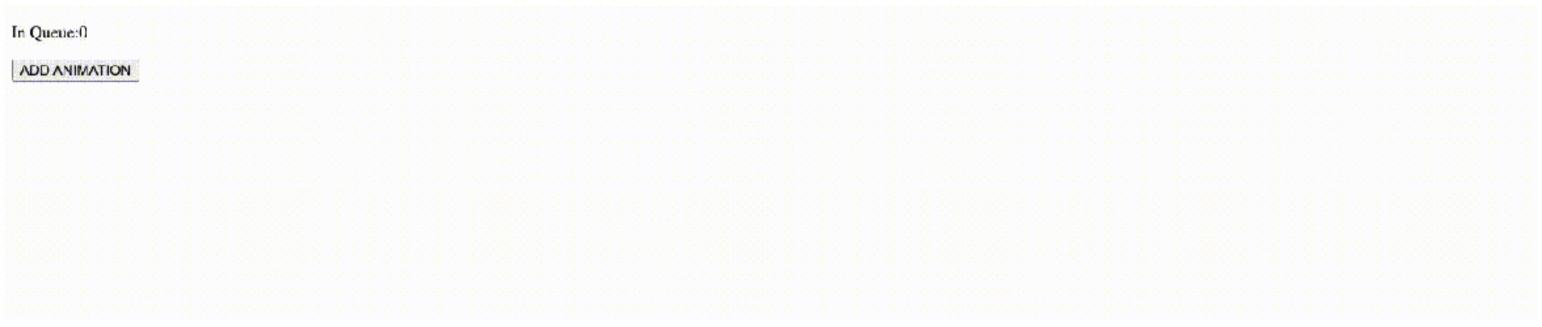
Copy

```
//generate loading bars
const generateLoadingBar = () => {
    //fragement
    const fragment = document.createDocumentFragment();

    //create a div elm
    const loadingBar = document.createElement("div");
    //apply styles
    //animation keyframes
    dynamicAnimation(
        "loadingBar",
        `
    0%{
        width: 0%;
    }
    100%{
        width: 100%;
    }`
    );
    loadingBar.style.height = "10px";
    loadingBar.style.backgroundColor = "Red";
    loadingBar.style.width = "0";
    loadingBar.style.marginBottom = "10px";
    loadingBar.style.animation = "loadingBar 3s forwards";

    //create shadow loading bar
    const shadowLoadingBar = document.createElement("div");
    //apply styles
    //animation keyframes
    dynamicAnimation(
        "shadowLoadingBar",
        `
    0%{
        width: 0%;
    }
    100%{
        width: 50%;
    }`
    );
    //it will be hidden
    shadowLoadingBar.style.height = "5px";
    shadowLoadingBar.style.backgroundColor = "green";
    shadowLoadingBar.style.width = "0";
    shadowLoadingBar.style.marginBottom = "10px";
    shadowLoadingBar.style.animation = "shadowLoadingBar 1.5s forwards";

    //add the both the bars to the fragment
    fragment.appendChild(loadingBar);
    fragment.appendChild(shadowLoadingBar);

    //append the loading bar
    const entry = document.getElementById("entry");
    entry.appendChild(fragment);

    //on animation end on shadowbar
    shadowLoadingBar.addEventListener("animationend", () => {
        //decrease the count
        updateCount(-1);
        if (count > 0) {
             //generate the loading bar
            generateLoadingBar();
        }
    });

    //remove listener
    shadowLoadingBar.removeEventListener("animationend", () => {});
};
```

If you notice, I am creating two loading bars and adding them in fragments, and hard-coded the duration of the shadow bar `1.5` and width to `50%`. You can make it dynamic by using a simple mathematical calculation.

Also, I have kept the background of the shadow bar green and it is visible currently (just to show the working), but you can hide it.

Everything else remains the same.



Prepare for your **JavaScript Interview** practically on each Interview rounds and grab that job.

BEGIN LEARNING

Recommended Posts:

Dutch national flag problem
Different ways to get element by id in javascript
Share data between two browser window with JavaScript
How to find length of an array in javascript
Compare two array or object with JavaScript
Capture product visible on viewport when user stops scrolling
Credit card validation in javascript
Array with event listeners in JavaScript
Convert an array to string javascript
How to check if given object is array in javascript

Prev Next