WEB ANIMATION API:

Animation is like showing pictures in different fram of time. For Animation, we use Web Animation Api as it is simple and gives better performance.

TWO MODELS: TIME AND ANIMATION:

TIME:

It is a default time which starts when we load our page, we don’t have to do anything much for that.

ANIMATION:

Animate method is used. It takes 2 parameters, one is keyframes(at what time what would show, like at 70% of time this shows, at 80% of given time this shows) and second parameter is the time(when animation start, how long it will run or when it ends). In simple, animation is transforming, changing or updating the property of element (like background color, radius, height, width etc). The given duration in animate method is distributed among all the frames.

document.querySelector(".shape")

document.querySelector is used to select property to which we want to apply animation

window.addEventListener("load", function ()

window.addEventListener is used to manage when animation done, like on page load, or on button click etc.

CODE FOR ANIMATION WHEN PAGE LOAD:

  <script>

        window.addEventListener("load", function (){

            var shapeElement = document.querySelector(".shape");//selects shape property

            var frames = [

                {background: "red", transform:"translateX(0px)"},

                {background: "blue"},

                {background: "yellow", transform: "translateX(600px)"}

            ];//define frames

            var timing = {

                duration: 2000,

                iterations: Infinity,

                direction: "alternate"

            }//define timing

            shapeElement.animate(frames, timing);//pass two parameters to animate method and app on shap

        })

    </script>

   <div class="shape">

    </div>

CONTROL WHEN ANIMATION STARTS, STOPS etc:

           var shapeAnimate = shapeElement.animate(frames, timing);//pass two parameters to animate method and app on shap

            document.querySelector(".play").addEventListener("click", function(){

                shapeAnimate.play();

            });

            document.querySelector(".pause").addEventListener("click", function(){

                shapeAnimate.pause();

            });

            document.querySelector(".reverse").addEventListener("click", function(){

                shapeAnimate.reverse();

            });

            document.querySelector(".updateplaybackrate").addEventListener("click", function(){

                shapeAnimate.play();

            });

        });

The above will select play, pause reverese etc property and apply any given function on button click, that function can be to play animation, stop animation etc

   <button class="play">Play</button>

    <button class="pause">Pause</button>

    <button class="reverse">Reverse</button>

    <button class="updateplaybackrate">updateplaybackrate</button>

Animation in React Process:

useRef hook

Installation:

$ yarn add @wellyshen/use-web-animations

# or

$ npm install --save @wellyshen/use-web-animations

useRef gives you the access of dom element. In react, all of our elements are in the virtual dom and we don’t have access of them directly or their properties or functions etc.

useRef(null) -> it creates reference and we can store it in any of the const

const inputE1 = useRef(null);

Remember: useState update element by doing rerendering, where as useRef give access to element properties of dom elements and it does not perform rerendring

To access property of element, we first create useRef reference and store it in some variable. Then we can use any of the properties using:

variablename.current.propery

  useEffect(()=>{

    inputE1.current.focus();

  })

//using focus property

then we can provide store const name in any of the components tag in which we want to use useRef and it will access its property from dom element by calling useRef

    <div className="App">

      Enter name:

      <input type="text" ref={inputE1}/>

    </div>

Incase of animations, to apply animation on particular element, we need to access its properties or class, we can use useRef for that.

onReady and onFinish not supported, onUpdate will work fine

useWebAnimations is the function which required an object

In useWebanimations, we can provide keyframes and timing and it returns ref playState and getAnimation. The element or component or tag to which we want to apply animation,

we will pass ref={ref} in that tag

Playstate:

playState gives you the information of what the state currently is

{playState} -> like running, finished, pause ete….

  const {ref, playState, getAnimation} = useWebAnimations({

    keyframes: [

      {transform: "translateX(0px)", background: "red"},

      {transform: "translateX(500px)", background: "yellow"}

    ],

    timing: {

      duration: 3000,

      iterations: Infinity,

      direction: "alternate",

      easing: "ease-in-out",

    }

  });

  return (

    <div>

      <div ref={ref} style={{background: "red", width: "100px", height: "50px"}}>

      Hello World {playState}

      </div>

    </div>

getAnimation:

it give access to different function like play pause etc. It is used to control the animation object.

 return (

    <div>

      <div ref={ref} style={{background: "red", width: "100px", height: "50px"}}>

      Hello World {playState}

      </div>

      <div>

        <button onClick={()=>{

          getAnimation().play();

        }}>

          Play

        </button>

        <button onClick={()=>{

          getAnimation().pause();

        }}>

          Pause

        </button>

      </div>

onUpdate will run on each frame and give all the details of the frame, like playState, animate, animation etc.

  const {ref, playState, getAnimation} = useWebAnimations({

    keyframes: [

      {transform: "translateX(0px)", background: "red"},

      {transform: "translateX(500px)", background: "yellow"}

    ],

    timing: {

      duration: 3000,

      iterations: Infinity,

      direction: "alternate",

      easing: "ease-in-out",

    },

    onUpdate: ({playState, animate, animation}) => {

      console.log("Play State ", playState);

      console.log("animate ", animate);

      console.log("animation ", animation);

    }

//we can view the result of onUpdate on console

We can add many builtin animations as we want. From <https://animate.style/>

The process of adding them is like

First importing:

import useWebAnimations, {bounce} from '@wellyshen/use-web-animations';

Like I am importing bounce css

Way of use with useRef():

function App() {

  const {ref, playState} = useWebAnimations({...bounce});

Now we simply pass ref.

  return (

    <div>

      <div ref={ref} style={{background: "red", width: "100px", height: "50px"}}>

      Hello World {playState}

      </div>

    </div>

  );

}