**JS and CSS Clock**

We are going to put a specific rotate according on each hand

<div class="hand hour-hand"></div>

<div class="hand min-hand"></div>

<div class="hand second-hand"></div>

If you go into the dev tools and look up one of the divs and look into the styles of that div

On element style you can test and insert transform: rotate 20 degrees and see that the div you chose will rotate 20 degreese however it does so from the midpoint of the element vs the right hand side of the line ( the middle of the clock)

element.style {

1.  transform: rotate(20deg);

}

What we are going to have to do is to go into the css and transform the origin which is where it will do the rotation off of

So we go into

.hand {

And put transform-origin: 100%;

.hand {

width:50**%**;

height:6**px**;

background:black;

position: *absolute*;

top:50**%**;

transform-origin: 100**%**;

this will mean that the point of rotation (the origin of rotation of the transform) will happen at the 100% mark.

The elements percentile reads from left to right so along the x axis. when we put 100% that would be the most right side which is currently the point we want to rotate the element.

(0% would be most left side, 50% would be the middle etc. )

Now also currently the hands are pointed towards 9 o’clock so to put them facing 12 o’clock we rotate them 90 degrees

top:50**%**;

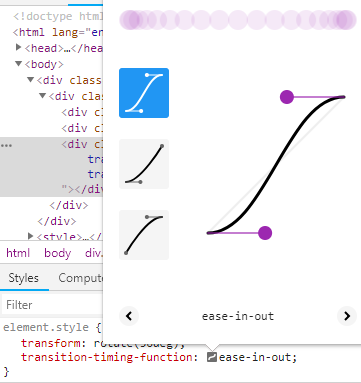
transform-origin: 100**%**;

transform: rotate(90**deg**);

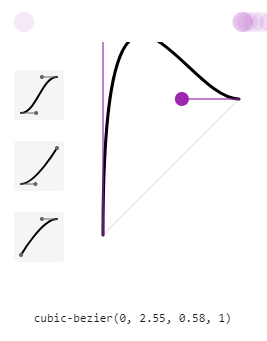
now we want to put a transition on the hands

we can play in the dev tools and put a transform rotate and see that is rotates correctly around the center of the clock but we can also put a transition-timing-function and use any of the pre selected options

in this case in dev tools used transition-timing-function: ease-in-out



We can manipulate the line by click an pulling on it and use that function



So we can copy that and insert it into our css and our transition will have that effect as if the hand launches to its new click but over shoots the mark then jerks back to its proper position imitating just like a real clock would

transition-timing-function: cubic-bezier(0, 2.55, 0.58, 1);

also going to make it even faster and change the transition from 0.5s to 0.05s

transition: *all* 0.05**s**;

now we can write a function to move the hands

so first we create a function called set date in our script tag and give a simple console log of hi to test

as well as if we want it to run every second(1000 mili seconds) we make a set interval

function setDate(){

**console**.log("hi");

}

setInterval(setDate, 1000);

and sure enough if we look in our console and refresh the page it says high every second

but we don’t want to console log hi we want to grab the date so

function setDate(){

const now **=** **new** Date();

const seconds **=** now.getSeconds();

**console**.log(seconds);

now we are seeing the seconds in the console but now we want to convert them into degrees

so we do seconds / 60 which gives us our percentage and them multiply times 360 so that will be our full

const secondsDegrees **=** ((seconds **/** 60) **\*** 360)

this next part in the video was confusing. (time 6:40 in vid)

he put a query selector outside the function and then put style inside the function.

(also this is ES6)

const secondHand **=** **document**.querySelector('.second-hand');

function setDate(){

const now **=** **new** Date();

const seconds **=** now.getSeconds();

**console**.log(seconds);

const secondsDegrees **=** ((seconds **/** 60) **\*** 360);

secondHand.**style**.transform **=** `rotate(${secondsDegrees}deg)`;

we are nearly their however the we see when 60 sec happens its not at the 12 o’clock mark its still at the old starting place so to fix this we add plus 90 degrees to the seconds hand to offset that 90 degrees we set in our css

const secondsDegrees **=** ((seconds **/** 60) **\*** 360) **+** 90;

now I duplicate the const secondhand and its style for min and hrs

const secondHand **=** **document**.querySelector('.second-hand');

const minsHand **=** **document**.querySelector('.min-hand');

const hourHand **=** **document**.querySelector('.hour-hand');

function setDate(){

const now **=** **new** Date();

const seconds **=** now.getSeconds();

const secondsDegrees **=** ((seconds **/** 60) **\*** 360) **+** 90;

secondHand.**style**.transform **=** `rotate(${secondsDegrees}deg)`;

const mins **=** now.getMinutes();

const minsDegrees **=** ((mins **/** 60) **\*** 360) **+** 90;

minsHand.**style**.transform **=** `rotate(${minsDegrees}deg)`;

const hour **=** now.getHours();

const hourDegrees **=** ((hour **/** 12) **\*** 360) **+** 90;

hourHand.**style**.transform **=** `rotate(${hourDegrees}deg)`;

}

setInterval(setDate, 1000);

Now it works however when it gets to the top it moves counter clockwise around the whole rotation to get to the next degree

We could tally the amount of degrees indefinitely (not suggested)

Or we could with JS temporarily take off the transition so that when it ticks to that next one it wont transition itself backwards and then reapply it after