Animation elective part I Now before we jump into animations, we are going to discuss two important types of css properties that will help us understand how to use animations better

Those are:

Transform Transition Transform The transform CSS property lets you rotate, scale, skew, or translate an element. It modifies the coordinate space of the CSS visual formatting model.

/* Keyword values */ transform: none;

/* Function values */ transform: rotate(0.5turn); transform: rotateX(10deg); transform: rotateY(10deg); transform: translate(12px, 50%); transform: translateX(2em); transform: translateY(3in); transform: scale(2, 0.5); transform: scaleX(2); transform: scaleY(0.5);

/* Multiple function values */ transform: translateX(10px) rotate(10deg) translateY(5px); Transform Transform functions - MDN

Transition CSS transitions provide a way to control animation speed when changing CSS properties. Instead of having property changes take effect immediately, you can cause the changes in a property to take place over a period of time. For example, if you change the color of an element from white to black, usually the change is instantaneous. With CSS transitions enabled, changes occur at time intervals that follow an acceleration curve, all of which can be customized.

CSS transitions let you decide which properties to animate (by listing them explicitly), when the animation will start (by setting a delay), how long the transition will last (by setting a duration), and how the transition will run (by defining a timing function, e.g. linearly or quick at the beginning, slow at the end).

You can use the transition shorthand or provide individual properties as well

div { transition: ; } /* another example is */ .box { border-style: solid; border-width: 1px; display: block; width: 100px; height: 100px; background-color: #0000FF; transition: width 2s, height 2s, background-color 2s, transform 2s; }

.box:hover { background-color: #FFCCCC; width: 200px; height: 200px; transform: rotate(180deg); }

You can also provide the individual properties in the following manner

.box{

```
background-color: red;
transition-property: transform, background;
transition-timing-function: linear;
transition-duration: 1s;
transition-delay: 1s;
```

}

Now if you wanted to create your own timing function for transition you can do that by using cubic bezier functions

.box{

```
background-color: red;
transition-property: transform, background;
transition-timing-function: cubic-bezier(.07,.82,.78,.44);
transition-duration: 1s;
transition-delay: 1s;
```

} Cuber Bezier

Cubic bezier is nothnig but a curve / spline that is generated between two points, P1, and P2 in the following graph

Where in cubic-bezier(x1, y1, x2, y2)

x1, y1 are coordinates of the point P1 x2, y2 are coordinates of the point P2 The spline is interpolated from this and the rate of change can be calculated. (Who said maths was not used in UI engineering!)

Transition

Ok but wait. What about animation? By now, we have understood:

how we can tranform elements how we can transition elements from one state to another Well there is one more property called animation that we can use, which builds on top of the knowleddge we just learnt.

Animation CSS animations make it possible to animate transitions from one CSS style configuration to another. Animations consist of two components, a style describing the CSS animation and a set of keyframes that indicate the start and end states of the animation’s style, as well as possible intermediate waypoints.

We saw two keywords, animation and keyframes

animation is the property that we can use (similar to transition, it is a shorthand) keyframes represent how the animation should behave, a bit similar to the cubic bezier or the timing functions we disucssed earlier, but a bit more controlled assuming an animation is divided into stages between 0 - 100% we can define points at any time between the start and end as well, and define how the property should animate if we just have two, that is the start and end, we can just use from and to animation @keyframe

There are three key advantages to CSS animations over traditional script-driven animation techniques:

They’re easy to use for simple animations; you can create them without even having to know JavaScript. The animations run well, even under moderate system load. Simple animations can often perform poorly in JavaScript. - The rendering engine can use frame-skipping and other techniques to keep the performance as smooth as possible. Letting the browser control the animation sequence lets the browser optimize performance and efficiency by, for example, reducing the update frequency of animations running in tabs that aren't currently visible. How do i use it? To create a CSS animation sequence, you style the element you want to animate with the animation property or its sub-properties. This lets you configure the timing, duration, and other details of how the animation sequence should progress. This does not configure the actual appearance of the animation, which is done using the @keyframes at-rule.

p { animation-duration: 3s; animation-name: slidein; }

@keyframes slidein { from { margin-left: 100%; width: 300%; }

to { margin-left: 0%; width: 100%; } } In this example the style for the element specifies that the animation should take 3 seconds to execute from start to finish, using the animation-duration property, and that the name of the @keyframes at-rule defining the keyframes for the animation sequence is named “slidein”.

@keyframes identifier { 0% { top: 0; left: 0; } 30% { top: 50px; } 60% { left: 50px; } 100% { top: 100px; left: 100%; } }

the keyframe selector can be either of these from | to | %