CSS Animations

Using CSS we can add some nice looking transitions and animations to our webpage, without any JS altogether. We can make very beautiful and appealing websites all by using CSS Animations.

To understand CSS Animation completely we need to master the following three CSS concepts in depth:

1. **Transforms**
2. **Transitions**
3. **Keyframes**

These 3 concepts together helps us to make any complex animation in CSS.

CSS Transforms

CSS transforms helps us to make changes to our html elements in a 2d plane by allowing them to moving from one position to other, resizing them, rotating them and stretching them. There are four main aspects of transforms:

1. Rotation
2. Scale (i.e. resizing)
3. Translate (i.e. moving)
4. Skew (i.e. stretching)

Translate - CSS Transform

CSS Translate helps us to move an element vertically or horizontally or both. To apply this transformation we can use the CSS `transform` rule. This rule takes multiple transformation actions space separated.
To move in X-Axis direction or we can say horizontally we can use translateX and to move in Y-Axis direction or we can say vertically we can use translateY.

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
```

```
        transform: translateX(120px); /* This will move the element with id
red, 120px in x-axis*/
}
```

OR

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: translateY(120px); /* This will move the element with id
red, 120px in y-axis*/
}
```

If we want to move in both x and y axis then we can give both the translations space separated to the transform property.

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: translateX(120px) translateY(120px); /* This will move
the element with id red, 120px in x-axis and y-axis both */
}
```

Now, there is a shorthand to apply the same behaviour of moving in both x-axis and y-axis. We can do it using the translate property.

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: translate(120px, 120px); /* This will move the element
with id red, 120px in x-axis and y-axis*/
}
```

The translate method takes two values, first is the displacement in x-axis and second is the displacement in y-axis. If we only give it one value like `translate(120px)` then it will only apply movement in 120px x-axis direction and assume y-axis displacement as 0.

Rotation - CSS Transform

Using CSS transform rules, we can rotate an element along with X axis, Y Axis or Z axis as well.

We can rotate our elements in the following few ways:

- rotateX - this will rotate the elements along with X axis
- rotateY - this will rotate the elements along with Y axis
- rotateZ - this will rotate the elements along with Z axis
- rotate - this can be used for a 2d rotation
- rotate3d - this can be used for a 3d rotation

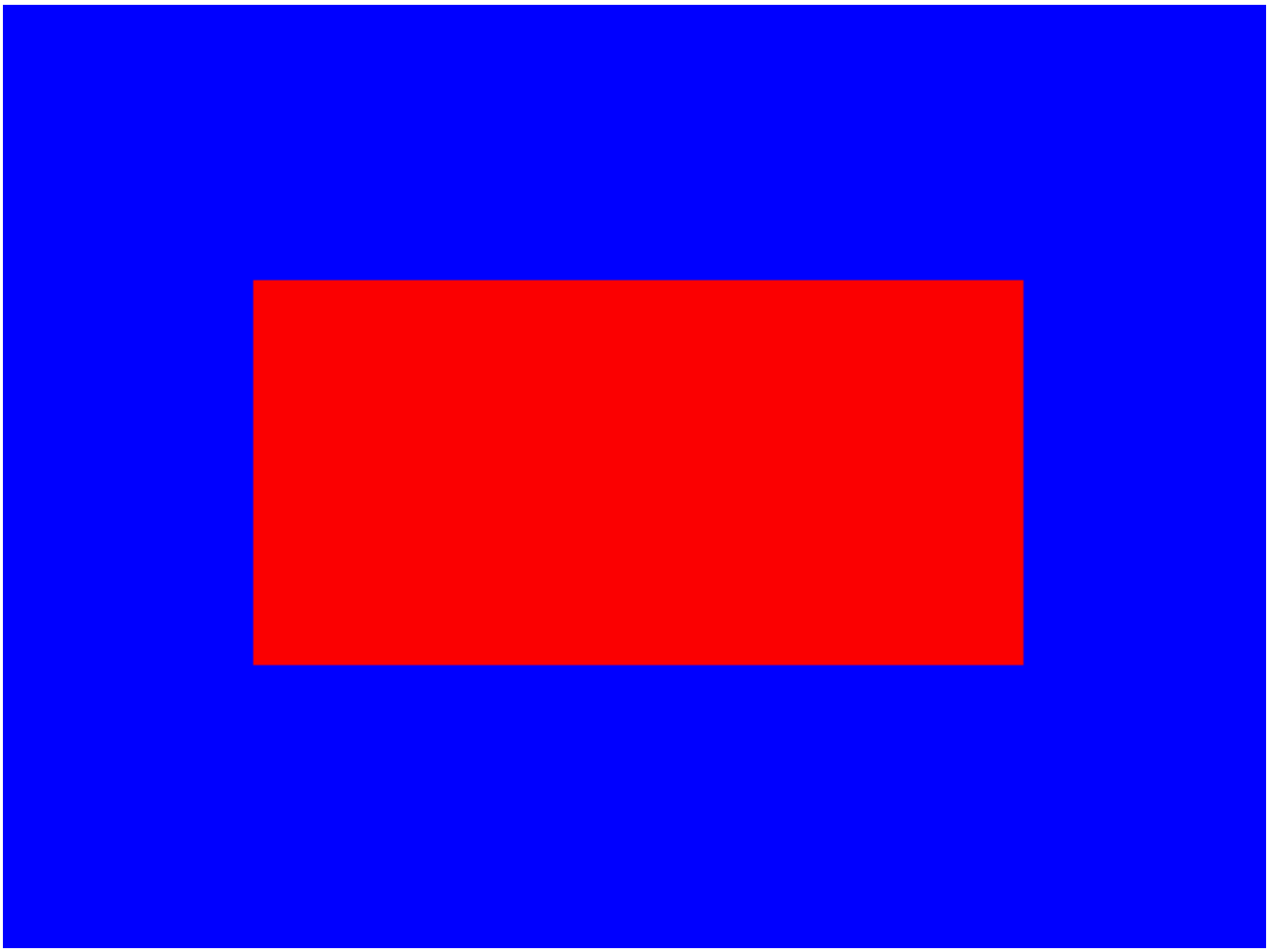
```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: rotate(45deg); /* This will rotate the element 45deg in
the 2d plane */
}
```

Scale - CSS Transform

If we want to resize the element i.e. make it bigger or smaller then we can use the scale property with CSS transform.
We can use the following properties to scale an element in different ways:

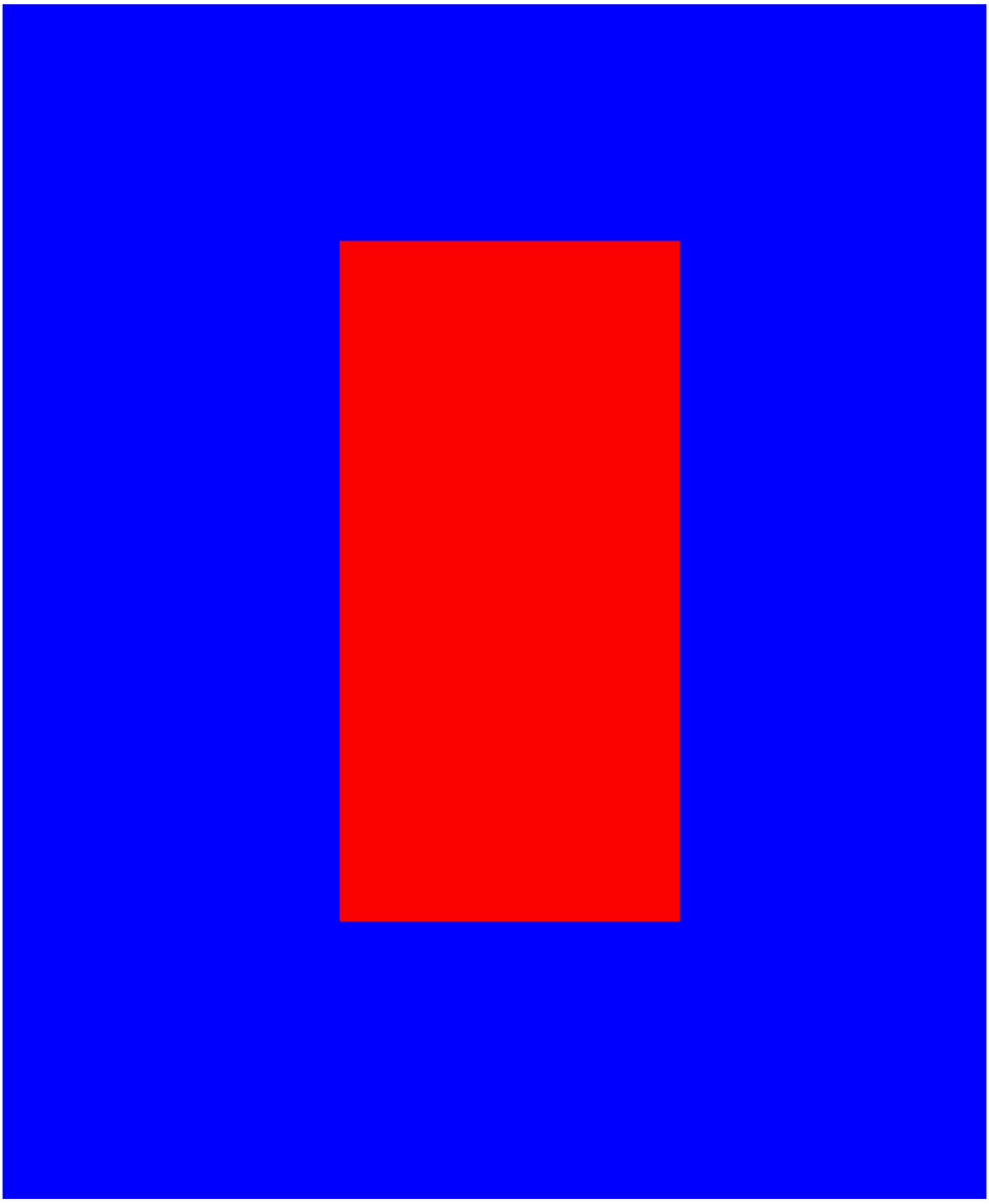
- scaleX - this will only make the element bigger or smaller along with X-Axis

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: scaleX(2); /* This will rotate the element 45deg in the
2d plane */
}
```



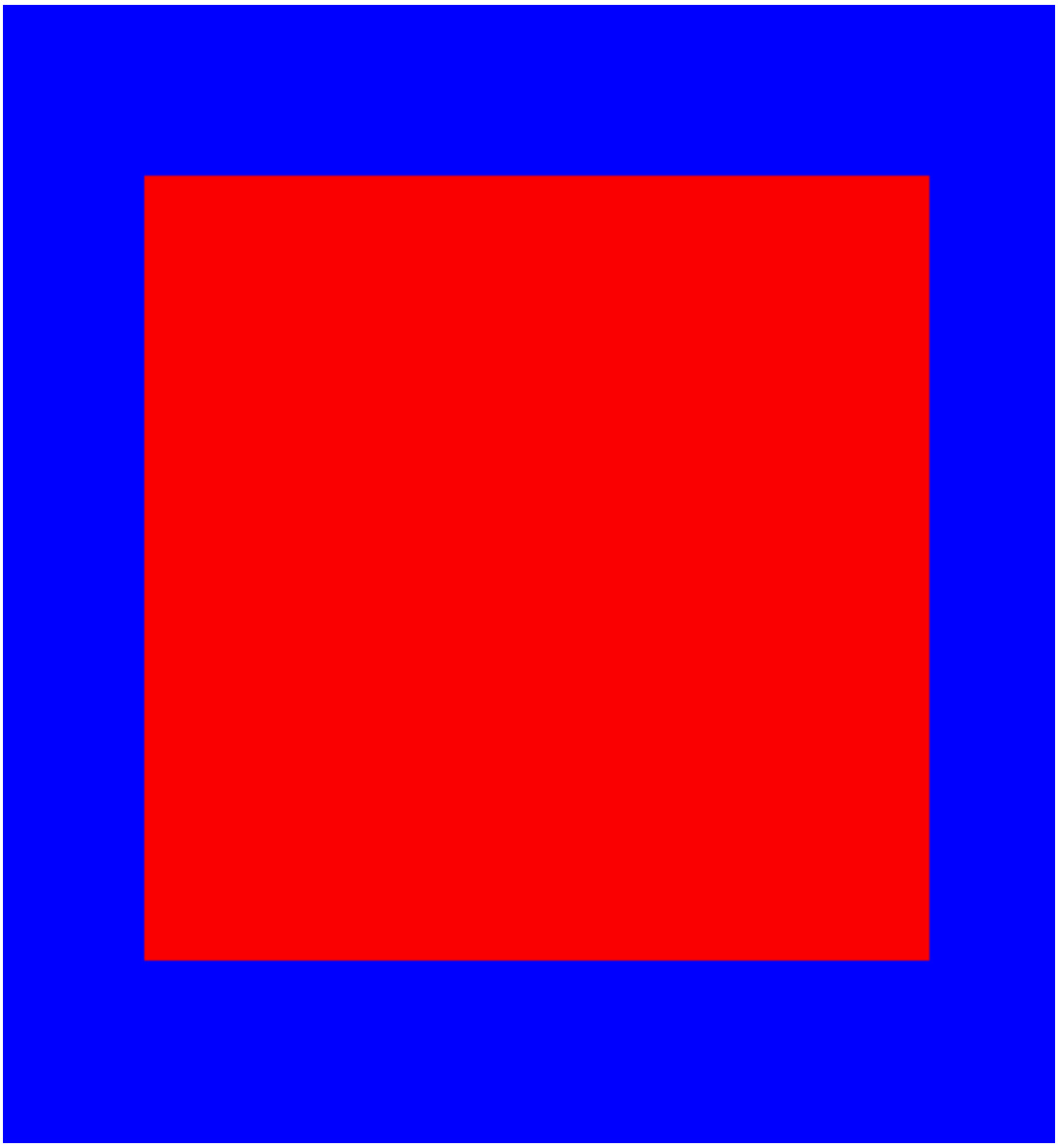
- scaleY - this will only make the element bigger or smaller along with Y -axis

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: scaleY(2); /* This will rotate the element 45deg in the
2d plane */
}
```



- scale - this will scale up or scale down elements on the 2d plane i.e. along both x and y axis

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: scale(2); /* This will rotate the element 45deg in the 2d
plane */
}
```



- scaleZ - this will scale up or scale down elements along the z axis
- scale3d - 3d scaling

Skew - CSS Transform

If you want to tilt, or stretch your element along with a particular axis , then we can use skew property. We can skew with positive and negative value both in terms of degree.

```
#red {
        height: 100px;
        width: 100px;
        background-color: red;
        transform: skew(20deg, 20deg); /* This will skew the element 20deg
in the 2d plane */
}
```

Note:

To combine all the 2d transformations in a single method we can also use `matrix`.

```
transform: matrix(scaleX, skewY, skewX, scaleY, translateX, translateY);
```

Or we can just give all transformations space separated as well .

CSS Transitions

If there is any change happening on the web page due to CSS and we want o make it look smooth, then transitions are the way to go for.
So to enable smooth transitions using CSS transition we can do the following:

- transition-property - Using this CSS rule we can select which property transition we are targeting. This mainly helps to select a particular property and then with other CSS rules we can control the transition.
- transition-duration - It defines, how much time should it take to complete the transition altogether.
- transition-delay - It defines a specific time after which the transition effect should be applied.
- transition-timing-function - this helps us to decide how the transition properties change during a transition. This describes the distribution of the changes of transition across the allocated transition-duration. This can take one of the following options:
  - ease: transition starts slowly, accelerates in the middle and ends slowly
  - linear: same speed of transition stays throughout the time span
  - ease-in: we start slow but end very fast
  - ease-out : we start very fast and end slowly
  - ease-in-out: make sure that start and end is both slow.
  - steps-end: specify the transition with a constant beginning that ends instantly.
  - cubic-beizer: this is to define a custom curve for the transition function

```
#expand {

        width: 100px;
        height: 100px;
        background-color: orange;
        transition-property: width;
        transition-duration: 5s;
        transition-timing-function: ease-in-out;
```

```
        transition-delay: 1s;
}

#expand:hover {
        width: 600px;
}
```

Instead of using all these properties separately, we can use the `transition` CSS rule to put all of the things together.

```
transition: transition-property transition-duration transition-timings-function transition-delay;
```

CSS Animations

Animations helps us to provide smooth transitions and transformations for a longer period of time. These can enhance user experience at a great scale. With animations we can add visual interests to our websites.

Keyframe rules

In CSS we have a keyword @keyframe. This CSS rule defines the elements style at various point of time during the animation. We can decide how the CSS for an element should change across the period of time.

Keyframe are defined in terms of percentage. These percentage shows the part of animation execution time. The 0% is the start of the animation and 100% is the end of the animation. Technically keyframes contain the change in css that should happen when an animation is triggered.

```
@keyframe name-of-animation {
        0% {
                background-color: orange;
        }
        10% {
                background-color: red;
        }
        40% {
                ...
        }
        80% {
```

```
                ...
        }
        100% {
        }

}
```

CSS animation duration

This property helps us to define the time for which animation should go for. We can use the `animation-duration` rule to define it.

CSS Animation name

This property helps us to define an animation name and then use it create keyframes. We can use `animation-name` css rule to define an animation name and then what should happen in that animation we can write in the keyframes.

```
#box {

        width: 100px;

        height: 100px;

        background-color: red;

}



#box:hover {

        animation-name: change-color;

        animation-duration: 5s;

}



@keyframes change-color { /* same as the aniimation name */
```

```
        0% {
                background-color: red;
        }
        50% {
                background-color: green;
        }
        100% {
                background-color: blue;
        }
}
```

CSS Animation delay

The `animation-delay` property specifies the delay in time before the animation starts. It can be represented in seconds or milliseconds.

CSS animation iteration count

This specifies then number of time an animation needs to continue. We can define this count in terms of numbers or once or infinite.

```
#box:hover {
        animation-iteration-count: infinite;
}
```

CSS Animation direction

This specifies the direction in which keyframe changes need to execute in an animation. The normal execution flow takes the changes from 0% to 100% and then again 0% to 100%. We can change this flow.

- reverse - In reverse direction keyframe will execute from 100% to 0%.
- alternate - In alternate, we first go forward normally and then we go backwards and so on
- normal - In normal direction keyframe will execute from 0% to 100%
- alternate-reverse - Here working is same as alternate but we first go backwards and then forward. \

```
#box:hover {
        animation-direction: alternate;

}
```

Animation timing function\

This specifies how the transition should sail throughout it's execution span. It ios very similar to that of transition-timing-function.

Animation property

This is a shorthand to put everything above in one place for animation.

```
#box:hover {
        animation: animation-name animation-duration animation-timing-
function animation-delay animation-iteration-count;
}
```