upGrad

#RahoAmbitious

Full Stack Software Development

upGrad

Course: Introduction to Web Development

Lecture On: CSS3 and CSS Animations

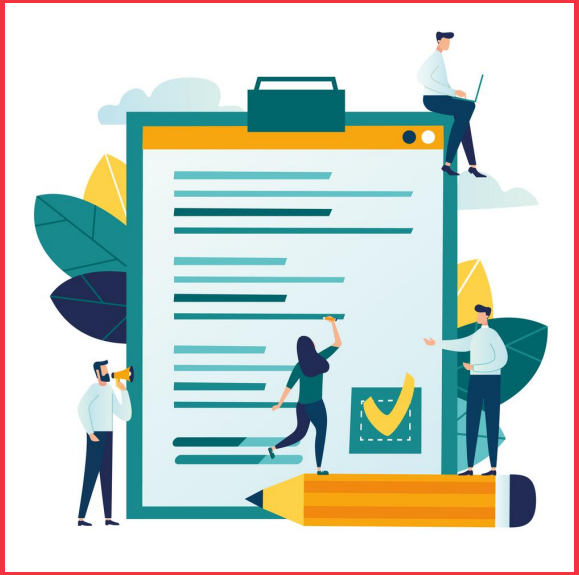
Instructor: Siddhesh Prabhugaonkar

In the previous class, we covered…

- What is flexbox and why is it the most sought-after CSS concept?
- How to place elements in a grid using CSS?

Today's Agenda

- Introduction to CSS3 and its new properties?
- CSS Transform
- CSS Transition
- CSS Animation



CSS3

Border Image and Multiple Background Images

Using the **border-image** property, you can assign an image to the border.

```
div {
   border-image-source: url(‘design.png’);
   border-image-repeat: round;
   border-image-width: 20px;
   border-image-slice: 30;
}
```

Border Image and Multiple Background Images

In CSS3, you can now add multiple backgrounds using the original background properties, but now, you can add a comma (‘,’) to add multiple values

```
div {
   background-image: url(‘oranges.png’), url(‘gallery.png’);
   background-position: left top, right top;
}
```

Gradients

Gradients are a combination of two or more colors. CSS3 introduces the following two types of gradients: linear and radial.

For a linear gradient, you can use the **linear-gradient** property. It takes multiple colors in arguments separated by commas (,). If the first argument is not specified, then, by default, it takes 'to right' as the direction.

```
div {
   background: linear-gradient(blue, red);
}
```

For a radial gradient, you can use the **radial-gradient** property. It takes multiple colors in arguments separated by comma (,). Every color also has a percentage of its contribution to the gradient and the default position for radial-gradient is center

```
div {
   background: radial-gradient(red 60%, blue 40%);
}
```

CSS Transform

CSS Transform

upGrad

- CSS transform allow you to move, rotate, scale or skew elements.
- With CSS transforms, you can do 2D transforms (along two directions) or 3D transforms (along three directions).
- In 2D transforms, you can use the following properties: **translate()**, **translateX(), translateY(), rotate()**, **scaleX()**, **scaleY()**, **scale()**, **skewX()**, **skewY()**, **skew()**, and **matrix().**
- Similarly, in 3D transforms, in addition to the properties in 2D, you can also use the following properties: **translate3d()**, **translateZ()**, **rotate()**, **rotateZ()**, **rotate3d()**, **scaleZ()**, **scale3d()**, **matrix3d()** and **perspective()**.

upGrad

2D Transform

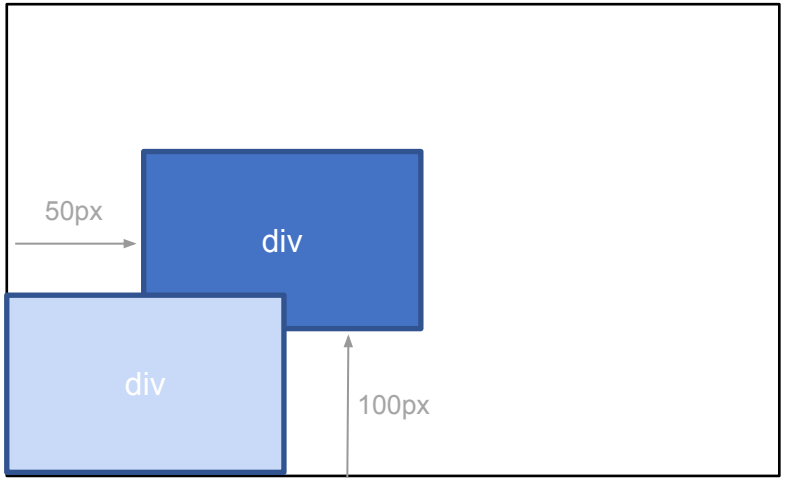
CSS Transform

upGrad

2D Transform: translate()

- The **translate()** property moves an element from its current position, according to the parameters given for x-axis and y-axis.
- The first argument is the reposition length on x-axis and the second argument is the reposition length on y-axis.
- Positive value of the first argument will make the box move to right and the negative value will move to left.
- Similarly, positive value of the second argument will make the box move to top but the negative value will make it move to bottom.

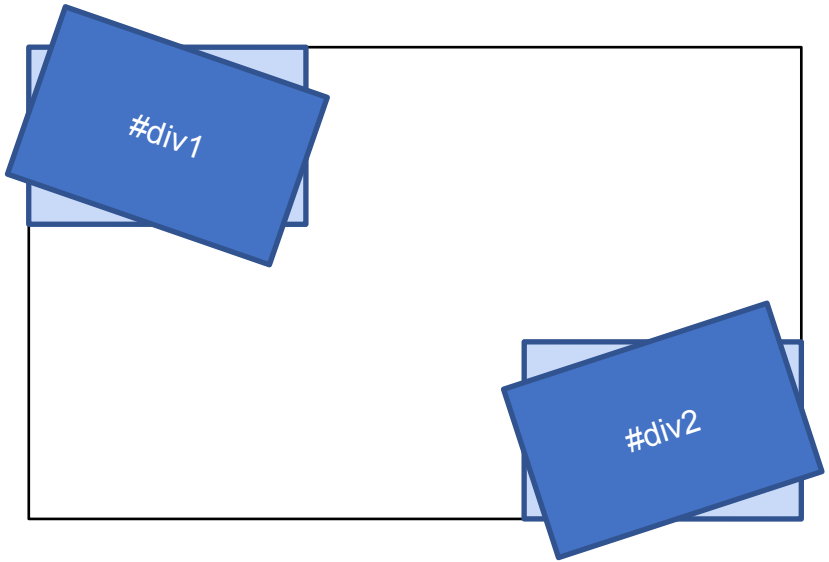
```
div {
    transform: translate(50px, 100px);
}
```



2D Transform: rotate()

The **rotate()** property rotates an element from its current position. A positive value will rotate it clockwise, whereas a negative value will rotate it anti-clockwise.

```
#div1 {
    transform: rotate(20deg);
}

#div2 {
    transform: rotate(-20deg);
}
```



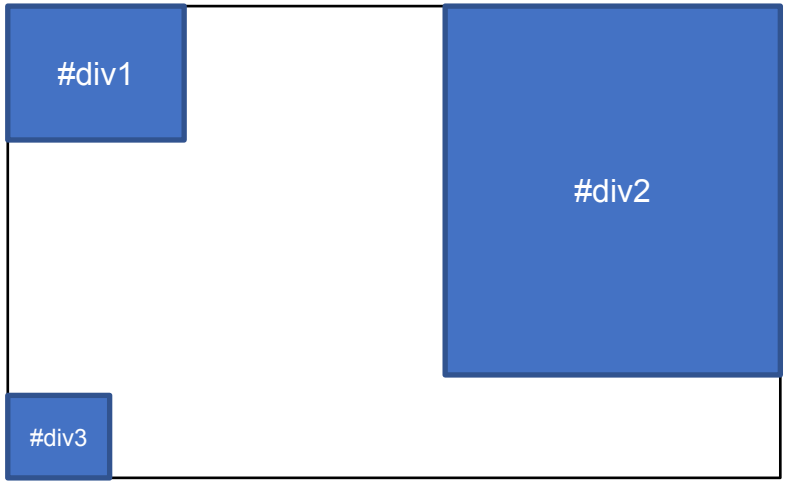
CSS Transform

upGrad

2D Transform: scale()

The **scale()** property increases or decreases the size of an element based on the given parameter. It takes two arguments; the first argument corresponds to the width, and the second one corresponds to the height. For instance, *scale(2, 3)* will increase the width of the element to two times its original width and increase the height to three times its original height. However, *scale(0.5, 0.5)* will halve the height and the width of an element. Alternatively, you can use **scaleX()** and **scaleY()** to change the width and the height of an element, respectively.

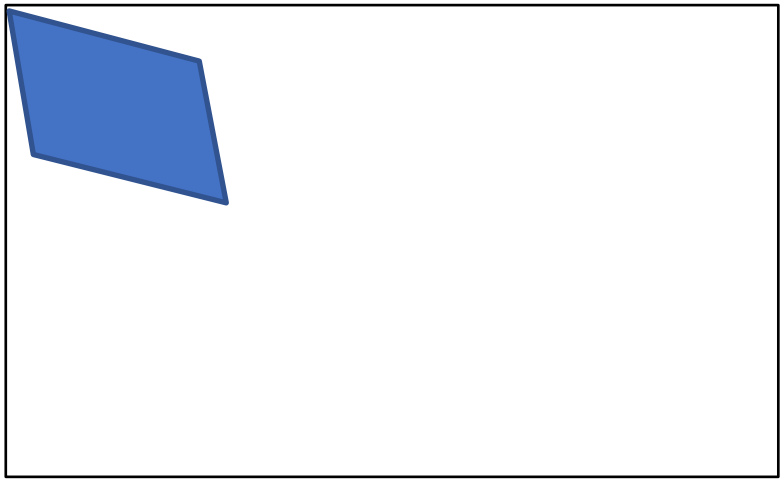
```
#div1 {
    transform: scale(1, 1);
}

#div2 {
    transform: scale(2, 3);
}

#div3 {
    transform: scale(0.5, 0.5);
}
```



2D Transform: skew()

The **skew()** property skews an element, i.e., turns it into an oblique angle, based on the given parameters. It takes two arguments: the first argument corresponds to the X-axis, and the second one corresponds to the Y-axis. For instance, *skew(20deg, 30deg)* will skew the element by 20 degrees along the X axis and 30 degrees along the Y axis. Alternatively, you can use **skewX()** and **skewY()** to skew elements along the X axis and the Y axis, respectively.

```
div {
    transform: skew(20deg, 30deg);
}
```



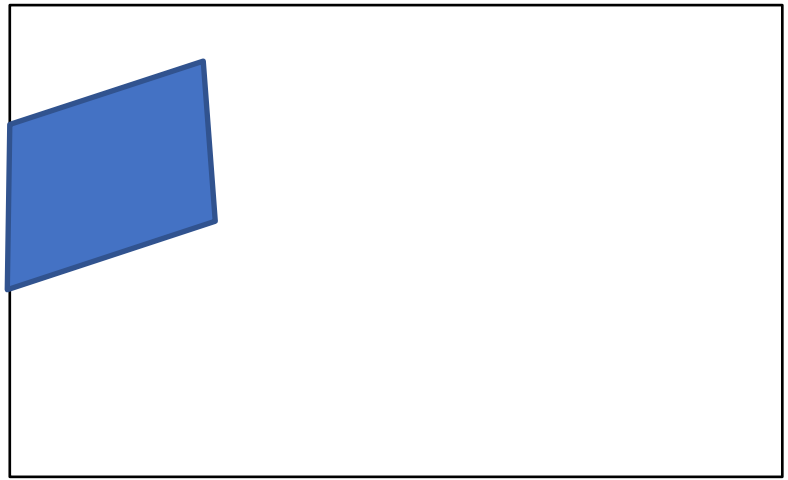
CSS Transform

upGrad

2D Transform: matrix()

The **matrix()** property combines all the 2D transforms into one. The *matrix()* property takes six parameters in the following order: *scaleX(), skewY(), skewX(), scaleY(), translateX()* and *translateY().*

```
div {
    transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```



Poll 1 (15 Sec)

What will be the output of the following CSS property?

```
div {
    transform: rotate(90deg);
}
```

1. It will rotate the element by 90 degrees in an anti-clockwise direction.
2. It will rotate the element by 90 degrees in a clockwise direction.
3. Neither (a) nor (b)

Poll 1 (Answer)

What will be the output of the following CSS property?

```
div {
    transform: rotate(90deg);
}
```

1. It will rotate the element by 90 degrees in an anti-clockwise direction.
2. **It will rotate the element by 90 degrees in a clockwise direction.**
3. Neither (a) nor (b)

Poll 2 (15 Sec)

What will be the output of the CSS code given here?

```
div {
  transform:
translate(100px, 200px);
}
```

1. The element will move 100px to the right and 200px downwards with respect to its current position.
2. The element will move 100px downwards and 200px to the right with respect to its current position.
3. The element will move 100px to the right and 200px upwards with respect to its current position.
4. The element will move 100px upwards and 200px to the left with respect to its current position.

Poll 2 (Answer)

What will be the output of the CSS code given here?

```
div {
  transform:
translate(100px, 200px);
}
```

1. **The element will move 100px to the right and 200px downwards with respect to its current position.**
2. The element will move 100px downwards and 200px to the right with respect to its current position.
3. The element will move 100px to the right and 200px upwards with respect to its current position.
4. The element will move 100px upwards and 200px to the left with respect to its current position.

CSS Transitions

CSS Transitions

upGrad

- The CSS **transition** property allows you to animate an element between two states over a certain duration of time. It is a shorthand property.
- For the transition property to work effectively, you need to specify two things **mandatorily**: **the CSS property** to which you want to add an effect and **the duration**. If the duration is not added, the transition will not work.
- For instance, let's assume that the element <*div*> had a width of 300px initially. When you hover the cursor over the element, the width changes to 600px. Adding a *transition: width 2s*, when you move the cursor over the element, the width of the div changes smoothly in 2 seconds. Similarly, when you move the cursor away from the element, the width reverts to 300px in 2 seconds.
- You can add multiple CSS properties to the *transition* property, but they should be separated by comma.

```
div {
    width: 300px;
    height: 100px;
    background-color: blue;
    transition: width 2s linear 2s;
}
div:hover {
    width: 600px;
}
```

You can find the list of animated properties here

CSS Transitions

upGrad

Properties of Transition

- You have already seen a shorthand property for **transition,** which has a **transition-property** and a **transition-duration**. Given below is a list of all the properties for transition that can be specified individually.

1. **transition-property:** This property specifies which CSS properties should undergo transition. If the value is given as ***'all'***, which is also the default value, all the properties, listed inside the element, that can transition will undergo transition.

```
transition-property: height, width, font-size;
```

1. **transition-duration:** This property specifies the duration for which the transition should last. The default value is 0s (s for seconds), meaning that no transition will take place.

```
transition-duration: 2s, 3s;
```

2. **transition-delay**: This property specifies the delay after which a transition should begin.The default is 0s (s for seconds), meaning that the transition will start immediately on specified action.

```
transition-delay: 4s;
```

3. **transition-timing-function:** This property specifies the speed curve of the transition effect. The values possible are as follows: *ease (default value: slow start to the transition, and then fast and end slow), linear (start and end the transition with the same speed), ease-in (slow start), ease-out (slow end), ease-in-out (slow start and slow end), etc.*

```
transition-timing-function: ease;
```

Poll 3 (15 Sec)

Which of the following values can be added in the ***transition-timing-function*** property? (More than one option may be correct.)

1. linear
2. ease-in
3. ease-out
4. ease

Poll 3 (Answer)

Which of the following values can be added in the ***transition-timing-function*** property? (More than one option may be correct.)

1. **linear**
2. **ease-in**
3. **ease-out**
4. **ease**

Poll 4 (15 Sec)

Which of the following CSS properties allows you to translate, scale, skew or rotate an element?

1. transition
2. transform
3. translate
4. rotate

Poll 4 (Answer)

Which of the following CSS properties allows you to translate, scale, skew or rotate an element?

1. transition
2. **transform**
3. translate
4. rotate

CSS Animations

Introduction to Animation and the @keyframes Property

- The CSS **animation** property allows you to animate elements using CSS.
- The major difference between *animation* and *transition* is that in transition, you can only **define how a change will occur**, but you cannot change the value of the CSS properties. In animation, you can change the value of the CSS properties inside the *keyframes*.
- The *keyframes* specify the new styles of an element and describe how it should gradually change. As shown in the two examples given below, you can either specify the two extreme ends *from* and *to*, or you can specify individual percentage state of the animation. For instance, if you want the animation to run for 20 seconds, then 50% would imply the point in time when 10 seconds have elapsed after the start of the animation.
- Using animation, you can **gradually** change one or multiple styles of an element. To use CSS animations, you first need to specify a *keyframe* and then call that particular keyframe using the *animation* property.

```
@keyframes colorChange {
    from { background-color: red; }
    20% { background-color: blue; }
    50% { background-color: yellow; }
    70% { background-color: green; }
    to { background-color: purple; }
}
```

Animation Properties

After the **@keyframes** are specified, you can run an animation by calling the **animation-name**(the value for this will be the value given to the @keyframe) and **animation-duration** properties. If a duration is not specified, then the animation will not run.

1. **animation-delay:** This property specifies the delay after which an animation should begin.
2. **animation-iteration-count:** This property specifies the number of times the animation should run. Setting it to *infinite* will run the animation forever.
3. **animation-direction:** This property specifies the direction and the order of the animation. Essentially, you can run the animation from the start to the end or in the reverse direction, starting with 100% and ending with 0%.
4. **animation-timing-function:** This property specifies the speed curve of the animation.

```
@keyframes colorChange {
    0% { background-color: red; }
    50% { background-color: yellow; }
    100% { background-color: green; }
}

div {
    animation-name: colorChange;
    animation-duration: 4s;
    animation-delay: 2s;
    animation-iteration-count: 3;
    animation-direction: normal;
    animation-timing-function: ease;
}
```

Animation Shorthand Property

As with almost every major CSS property, CSS animations also have a shorthand property. The syntax for the **animation** property is as follows:

```
animation: animation-name animation-duration animation-timing-function animation-delay
animation-direction;
```

```
@keyframes colorChange {
    0% { background-color: red; }
    50% { background-color: yellow; }
    100% { background-color: green; }
}

div {
    animation: colorChange 2s ease 4s 10 normal forwards running;
}
```

- *Note that it is NOT mandatory to have all the eight values for the shorthand property. Usually, you will only need the first three values, which you can specify as animation: colorChange 2s ease.*

Poll 5 (15 Sec)

Which of the following is the default duration of an animation when not specified?

1. 1s
2. 0s
3. 2s
4. 5s

Poll 5 (Answer)

Which of the following is the default duration of an animation when not specified?

1. 1s
2. **0s**
3. 2s
4. 5s

Poll 6 (15 Sec)

What does the given CSS code specify?

```
@keyframes percentexample {
  0%   { background-color: pink; }
  25%  { background-color: purple; }
  50%  { background-color: grey; }
  100% { background-color: black; }
}
```

1. It specifies that the background-color property of the element will initially be pink, and then it will change into other colors when the animation is 25% complete, 50% complete, 75% complete and 100% complete, respectively.
2. The background-color of the element will not change; it will constantly remain pink.
3. The background-color of the element will not change; it will constantly remain black.

Poll 6 (Answer)

What does the given CSS code specify?

```
@keyframes percentexample {
  0%   { background-color: pink; }
  25%  { background-color: purple; }
  50%  { background-color: grey; }
  100% { background-color: black; }
}
```

1. **It specifies that the background-color property of the element will initially be pink, and then it will change into other colors when the animation is 25% complete, 50% complete, 75% complete and 100% complete, respectively.**
2. The background-color of the element will not change; it will constantly remain pink.
3. The background-color of the element will not change; it will constantly remain black.

Poll 7 (15 Sec)

What does the negative value of the animation-duration property specify?

1. The negative value has no effect and will start from 0s only.
2. The animation will start in a such way that it has already been played past N seconds.
3. A negative value is not allowed.
4. None of the above

Poll 7 (Answer)

What does the negative value of the animation-duration property specify?

1. The negative value has no effect and will start from 0s only.
2. The animation will start in a such way that it has already been played past N seconds.
3. **A negative value is not allowed.**
4. None of the above

Hands-On Exercise 1 (4 mins)

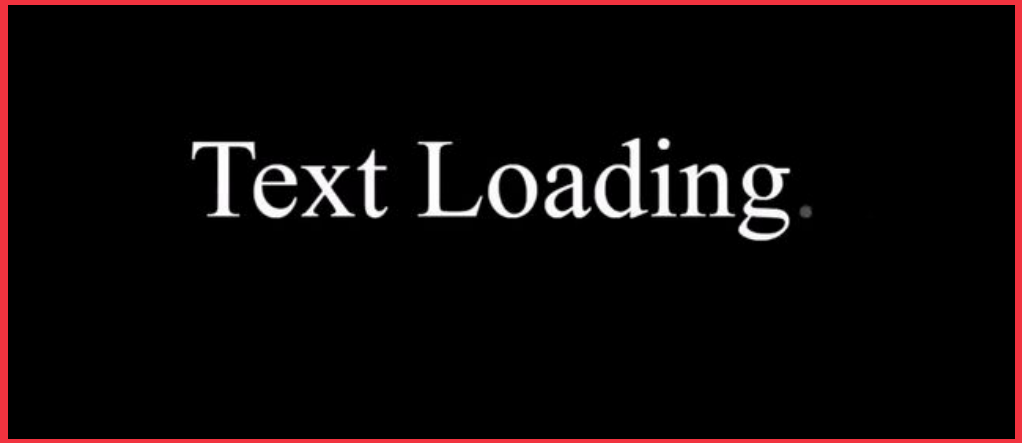
upGrad

Write the CSS code for the given HTML code such that the output should be as shown in the screenshot.

Note: You will have to use CSS animations to get the desired result.

Stub Code: here

Solution Code: here



Project Work

(We will add the flexbox to our project.)

You can refer to the solution here.

Key Takeaways from this class...

- CSS3 introduces new properties such as multiple backgrounds, gradients and text-based properties.
- Using CSS Transforms, you can move, rotate, scale and skew an element along X and Y axes or X, Y and Z axes.
- Using CSS Transitions, you can define how a change should occur on an element over a given duration.
- Using CSS Animations, you can cause an element to change over a given duration.

upGrad

Doubt Clearance (5 mins)

The following tasks are to be completed after today's session:

MCQs
Coding Questions
Project - Checkpoint 8

upGrad

In the next class, we will discuss...

- Responsive web design
- Bootstrap

upGrad

#RahoAmbitious

Thank you!