**Animation:-**

Following properties are used for animation:-

* @keyframes:- Holds styles of elements. Animation will change from current style to new style.
* animation-name:- specifies name of animation.
* animation-duration:- specifies time in seconds to perform animation.
* animation-delay:- specifies time in seconds to delay the animation.
* animation-iteration-count:- specifies how many times animation will be performed.
* animation-direction:- specifies direction of animation with values normal, reverse, alternate and alternate-reverse.
* animation-timing-function:-specifies speed of animation with values ease, linear, ease-in, ease-out, ease-in-out, and cubic-bezier(n,n,n,n)
* animation-fill-mode: specifies style for target
* animation element when animation is not playing with values none, forward, backwards and both.

**@keyframes:-**

Keyframes hold the style which are given to elements. When css style is specified inside @ keyframes rule animation will change from current style to new style. To get animation to work you must bind the animation to an element.

e.g. @keyframes abc

{

From{background-color:blue;}

To{background-color:red;}

}

Div

{

Height:100px;

Width:100px;

Background-color: blue;

Animation-name: abc;

Animation-duration: 4s;

}

Animation-duration property must be specified because without it animation will not occur. Instead of from and to we can also specify animation in % format.

e.g. @keyframes

{

0%{background-color: pink;}

25%{background-color: red;}

50%{background-color: yellow;}

100%{background-color: blue;}

}

We can also change the position of element.

e.g. @keyframes

{

0%{background-color:pink;left:0px;top:0px;}

25%{background-color:red;left:200px;top:0px;}

50%{background-color:yellow;left:200px;top:200px;}

75%{background-color:green;left:0px;top:200px;}

100%{background-color:blue;left:0px;top:0px;}

}

**animation-delay:-** This property is used to set timing to delay an animation.

e.g. animation-delay:2s;

**animation-iteration-count:-** This property is used to decide how many times animation should run.

e.g. animation-iteration-count:3;

**animation-direction:-** This property decides in which direction animation should be played. It’s values are:

* normal:- It is the default value. It plays animation in forward direction.
* Reverse:-It plays animation in backward direction.
* Alternate:-It plays animation first in forward direction and then in backward.
* Alternate-reverse:-It plays animation first in backward direction and then forward direction.

**animation-timing-function:-** This property specifies speed curve of animation. It has following values:

* ease - Specifies an animation with a slow start, then fast, then end slowly (this is default)
* linear - Specifies an animation with the same speed from start to end
* ease-in - Specifies an animation with a slow start
* ease-out - Specifies an animation with a slow end
* ease-in-out - Specifies an animation with a slow start and end
* cubic-bezier(n,n,n,n) - Lets you define your own values in a cubic-bezier function

e.g. #div1 {animation-timing-function: linear;}  
#div2 {animation-timing-function: ease;}  
#div3 {animation-timing-function: ease-in;}  
#div4 {animation-timing-function: ease-out;}  
#div5 {animation-timing-function: ease-in-out;}

**Transitions:-** This effect will change shape of element not the position. It has following properties:

* transition:- Specifies type of transition.
* transition-delay:- Specifies time in seconds to delay transition.
* transition-duration:- Specifies time in seconds to run transition.
* transition-timing-function:- Specifies speed in curve for the transition.

e.g. 1) To change single property

div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s;  
}

div:hover {  
  width: 300px;  
}

e.g. 2) To change Several effects

div {  
  width: 100px;  
  height: 100px;  
  background: red;  
  transition: width 2s height 2s;  
}

div:hover {  
  width: 300px;

height:300px;  
}

**transition-delay**:- It specifies a delay (in seconds) for the transition effect.

e.g. div

{  
  transition-delay: 1s;  
}

**transition-timing-function:-**It specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

* ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
* linear - specifies a transition effect with the same speed from start to end
* ease-in - specifies a transition effect with a slow start
* ease-out - specifies a transition effect with a slow end
* ease-in-out - specifies a transition effect with a slow start and end
* cubic-bezier(n,n,n,n) - lets you define your own values in a cubic-bezier function

e.g. #div1 {transition-timing-function: linear;}  
#div2 {transition-timing-function: ease;}  
#div3 {transition-timing-function: ease-in;}  
#div4 {transition-timing-function: ease-out;}  
#div5 {transition-timing-function: ease-in-out;}

**Transition + Transformation**

div {

width: 100px;

height: 100px;

background: red;

transition: width 2s, height 2s, transform 2s;

}

div:hover {

width: 300px;

height: 300px;

transform: rotate(180deg);

}

**2D Transformation:-**

With the CSS transform property you can use the following 2D transformation methods:

* translate()
* rotate()
* scaleX()
* scaleY()
* scale()
* skewX()
* skewY()
* skew()
* matrix()

translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

e.g. div {  
  transform: translate(50px, 100px);  
}

rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the <div> element clockwise with 20 degrees:

e.g. div {  
  transform: rotate(20deg);  
}

The following example rotates the <div> element anti clockwise with 20 degrees:

div {  
  transform: rotate(-20deg);  
}

scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the <div> element to be two times of its original width, and three times of its original height:

e.g. div {  
  transform: scale(2, 3);  
}

scaleX() method increases or decreases the width of an element.

e.g. div {  
  transform: scaleX(2);  
}

scaleY() method increases or decreases the height of an element.

div {  
  transform: scaleY(0.5);  
}

skewX() method skews an element along the X-axis by the given angle.

div {  
  transform: skewX(20deg);  
}

skewY() method skews an element along the Y-axis by the given angle.

div {  
  transform: skewY(20deg);  
}

skew() method skews an element along the X and Y-axis by the given angles.

div {  
  transform: skew(20deg, 10deg);  
}

matrix() method combines all the 2D transform methods into one.

The parameters are as follow: matrix(scaleX(),skewY(),skewX(),scaleY(),translateX(),translateY())

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

**3D transformations:-** following 3D transformation methods can be used with transform propert.

* rotateX()
* rotateY()
* rotateZ()

rotateX() method rotates an element around its X-axis at a given degree:

#myDiv {  
  transform: rotateX(150deg);  
}

rotateY() method rotates an element around its Y-axis at a given degree:

#myDiv {  
  transform: rotateY(150deg);  
}

rotateZ() method rotates an element around its Z-axis at a given degree:

#myDiv {  
  transform: rotateZ(90deg);  
}