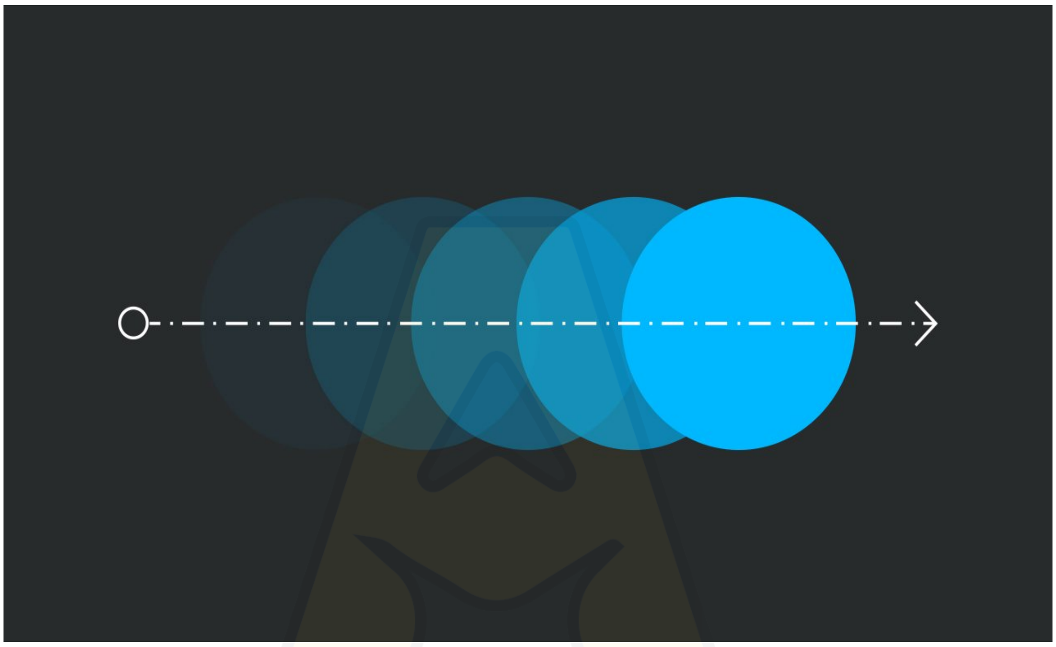
AIMERZ.ai
Aim.Act.Achieve

NOTES

Advance CSS -3

1. Animation

CSS animations allow you to animate the transition of CSS properties over time without using JavaScript. By defining keyframes that set the start, end, and intermediate states, you can control the duration, timing, and repetition of the animation, creating smooth visual effects and dynamic page elements.



- **Keyframes**: These define the start, end, and intermediate states of the animation.
- **Animation properties**: These control how the animation is executed, such as duration, delay, iteration count, and timing functions.

1.1 Keyframes:

Keyframe rule specifies the animation code, which animation is to be applied and how it will be performing.

Syntax:

```
Unset
@keyframes animationname {
          keyframes-selector {
                    css-styles;
          }
}
```

Elements of Keyframe:

1. **animationname:** This defines a specific name to the grouped properties you apply as an animation.
2. **keyframes-selector:** This will keep the % percentage value of the animation duration.

   Valid values applicable: [0-100%, from(same as 0%) , to(same as 100%)]
3. **CSS Styles:** All remaining styles will remain the same as applicable in css.

1.2 Animation Properties

CSS animation properties control the behaviour of animations applied to elements. They help define aspects such as the animation's duration, timing, delay, and more.

Here are the various animation properties given below:

1. **animation-name**
2. **animation-duration**
3. **animation-timing-function**
4. **animation-delay**
5. **animation-iteration-count**
6. **animation-direction**
7. **animation-fill-mode**
8. **animation-play-state**

Now let's implement each property with a demo example and expected output. Below mentioned all key properties with

1.2.1 animation-name :

Animation name property is applied to provide a name to an animation, it helps to reuse same named animation for multiple purposes in the web application. Now let's discuss the animation-name property with an example.

Example: Make a box and colour it to apply an animation using a keyframe.
HTML:

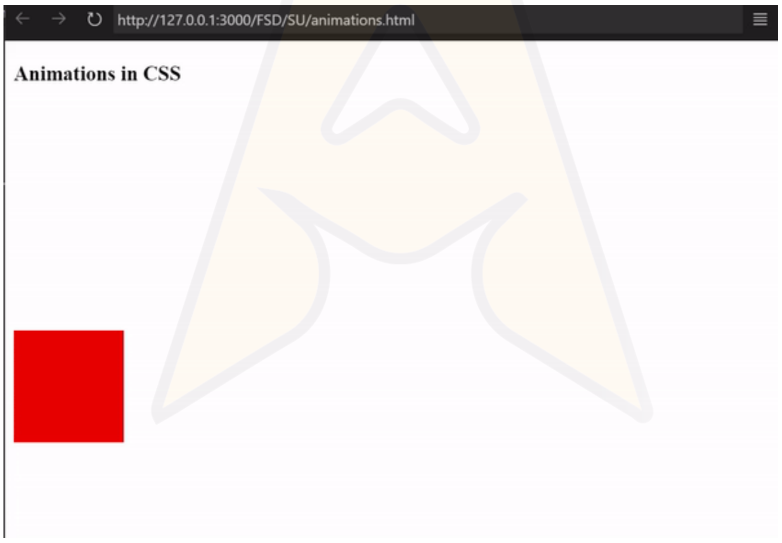
```
Unset
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <link rel="stylesheet" href="animation.css">
    </head>
    <body>
            <h3>Animations in CSS</h3>
            <div></div>
    </body>
```

```
</html>
```

CSS:

```
Unset
div {
    width: 100px;
    height: 100px;
    background: red;
    position: relative;
    animation-duration:2s;
    animation-name: mymove;
  }
  @keyframes mymove {
    from {top: 0px;}
    to {top: 200px;}
  }
```

OUTPUT:



Animations in CSS

1.2.2 animation-duration: Animation duration property is used to apply an animation for a specific duration, mostly it is specified in seconds. After completion of that duration the animation will be stopped.

Example: Create a dropping ball animation and apply the animation-duration property for a duration of 5 seconds.

HTML:

```
Unset
       <!DOCTYPE html>
       <html>
```

```
    <head>
        <meta charset="utf-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <title></title>
        <meta name="description" content="">
        <meta name="viewport" content="width=device-width,
initial-scale=1">
        <link rel="stylesheet" href="animation.css">
    </head>
    <body>
            <h3>Animations in CSS</h3>
            <div></div>
    </body>
</html>
```

CSS:

```
Unset
                 div {
    width: 100px;
    height: 100px;
    background: red;
    position: relative;
    border-radius: 100px;
    animation-name: mymove;
    animation-duration:5s;
  }
  @keyframes mymove {
  from {top: 0px;}
  to {top: 200px;}
}
```

OUTPUT:

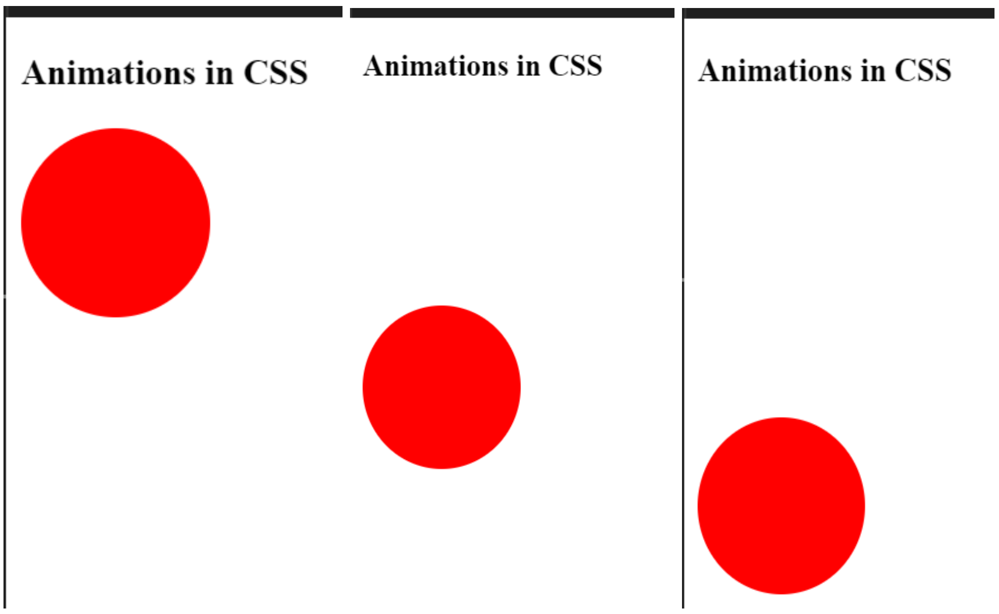


Fig: Animation of falling ball made using the css animation property.

1.2.3 animation-timing-function:

animation-timing-function controls the speed curve of the animation, letting you define how the animation transitions over time.

Here are some common values:

1.2.3.1 ease: starts slow, speeds up, then slows down at the end. This is the default.

1.2.3.2 linear: maintains a constant speed from start to finish.

1.2.3.3 ease-in: starts slow and speeds up.

1.2.3.4 ease-out: starts fast and slows down.

1.2.3.5 ease-in-out: starts slow, speeds up, then slows down again.

Example:

HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="box ease">Ease</div>
    <div class="box linear">Linear</div>
    <div class="box ease-in">Ease In</div>
    <div class="box ease-out">Ease Out</div>
    <div class="box ease-in-out">Ease In Out</div>
</body>
</html>
```

CSS:

```
Unset
.box{
    width: 100px;
    height: 100px;
    margin:8px;
    color: azure;
    background: red;
    position: relative;
    justify-content: center;
    align-content: center;
    border-radius: 100px;
    animation-name: mymove;
    animation-duration:5s;
    text-align: center;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
  }
  @keyframes mymove{
  from {left: 0px;}
  to {left: 200px;}
}


.ease {
  animation-timing-function: ease;
}

.linear {
  animation-timing-function: linear;
}

.ease-in {
  animation-timing-function: ease-in;
}

.ease-out {
  animation-timing-function: ease-out;
}

.ease-in-out {
  animation-timing-function: ease-in-out;
}
```

OUTPUT:

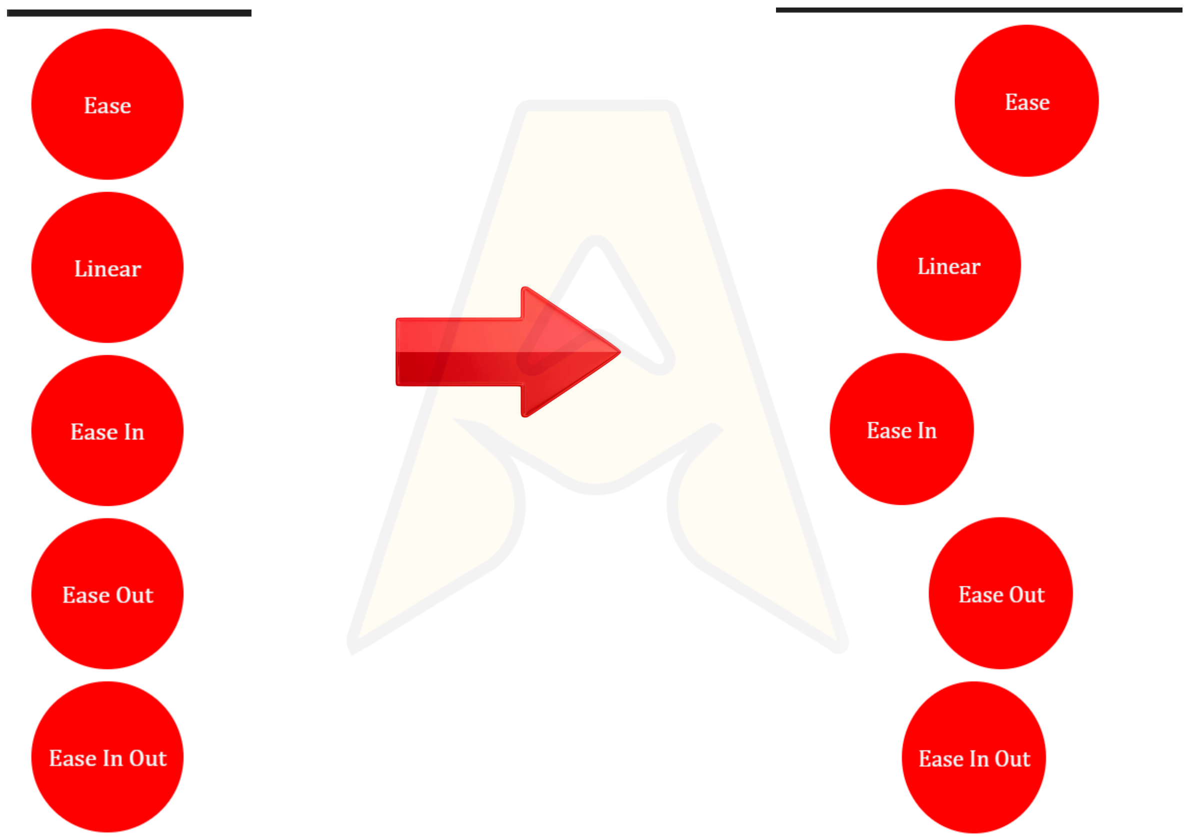


Fig: Demonstration of Animation timing function.

1.2.4 animation-delay:

The animation-delay CSS property specifies the amount of time to wait from applying the animation to an element before beginning to perform the animation. The animation can start later, immediately from its beginning, or immediately and partway through the animation.

Example: Apply an animation delay to a specific element to show how animation delay property works.

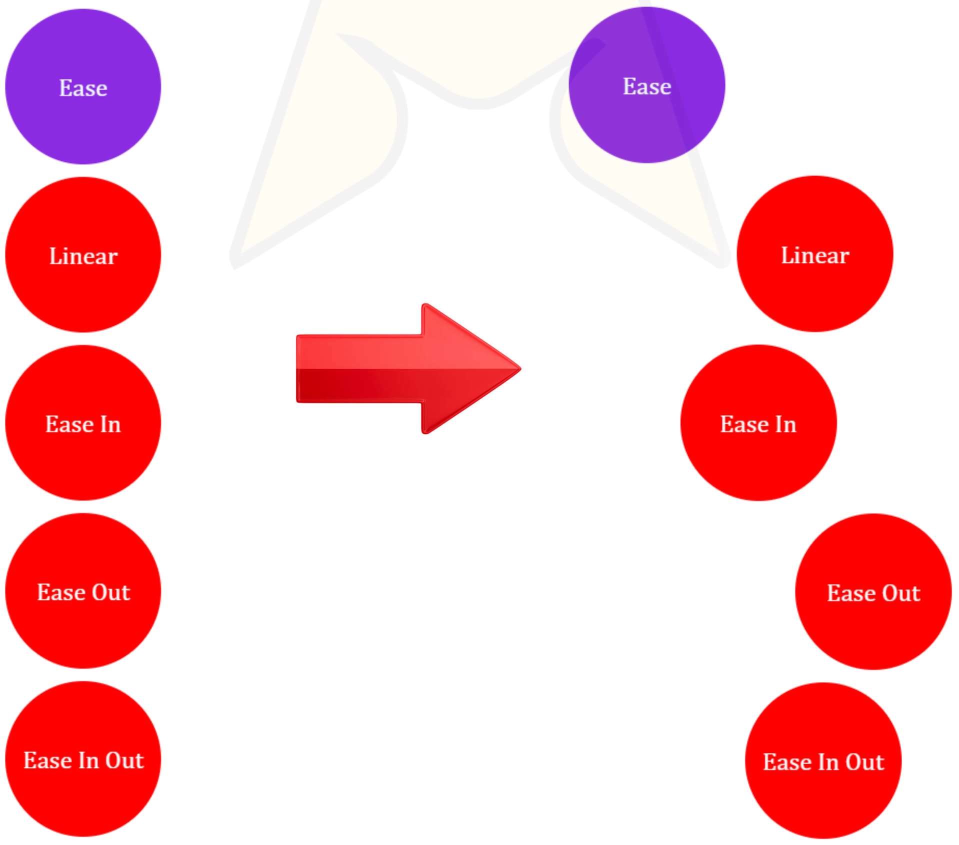
HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="box ease">Ease</div>
    <div class="box linear">Linear</div>
    <div class="box ease-in">Ease In</div>
    <div class="box ease-out">Ease Out</div>
    <div class="box ease-in-out">Ease In Out</div>
</body>
</html>
```

CSS:

```
Unset
.box{
    width: 100px;
    height: 100px;
    margin:8px;
    color: azure;
    background: red;
    position: relative;
    justify-content: center;
    align-content: center;
    border-radius: 100px;
    animation-name: mymove;
    animation-duration:5s;
    text-align: center;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
  }
  @keyframes mymove{
  from {left: 0px;}
  to {left: 200px;}
}


.ease {
   animation-timing-function: ease;
  background-color: blueviolet;
  animation-delay: 3s;
}

.linear {
   animation-timing-function: linear;
}

.ease-in {
   animation-timing-function: ease-in;
}

.ease-out {
   animation-timing-function: ease-out;
}

.ease-in-out {
   animation-timing-function: ease-in-out;
}
```

OUTPUT:



* Demonstration of animation delay, how a delay behaves on an element.

1.2.5 animation-iteration-count:

animation-iteration-count enables us to control how many times an animation repeats. You can set it to a number or make it loop endlessly with infinite.

Let us take an example to apply the animation-iteration-count property and explore its application areas.

Example: Create an animation and apply animation-iteration-count property

HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="box ease">Ease</div>
    <div class="box linear">Linear</div>
    <div class="box ease-in">Ease In</div>
    <div class="box ease-out">Ease Out</div>
    <div class="box ease-in-out">Ease In Out</div>
</body>
</html>
```

CSS:

```
Unset
.box{
    width: 100px;
    height: 100px;
    margin:8px;
    color: azure;
    background: red;
    position: relative;
    justify-content: center;
    align-content: center;
    border-radius: 100px;
    animation-name: mymove;
    animation-duration:5s;
    text-align: center;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
    animation-iteration-count: 2;
  }
  @keyframes mymove{
  from {left: 0px;}
  to {left: 200px;}
}


.ease {
  animation-timing-function: ease;
 background-color: blueviolet;
 animation-delay: 3s;
}

.linear {
  animation-timing-function: linear;
}

.ease-in {
  animation-timing-function: ease-in;
}

.ease-out {
  animation-timing-function: ease-out;
}

.ease-in-out {
  animation-timing-function: ease-in-out;
}
```

OUTPUT:

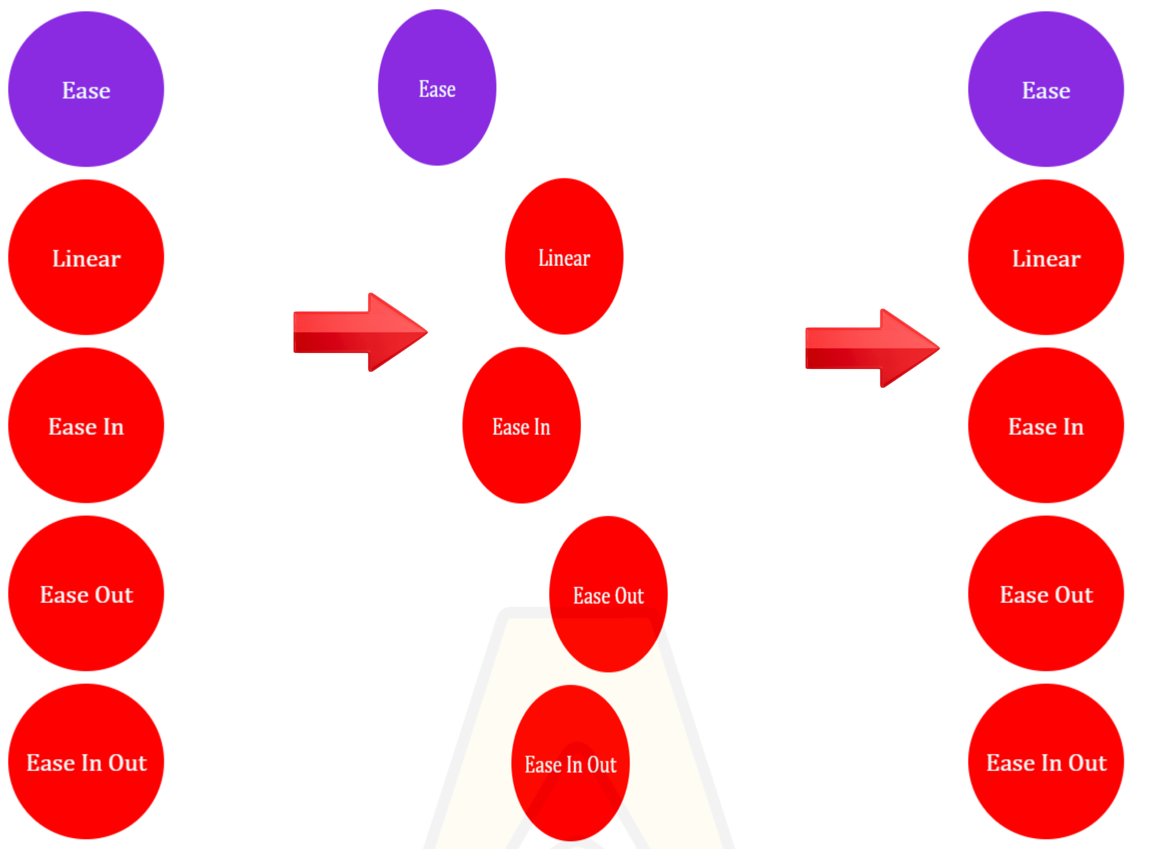


Fig: Initial Stage Fig: During Execution Fig: After Completing 2 counts

1.2.6. animation-direction:

Animation direction property is used to specify the direction on which we need to apply the specific animation as applicable, direction is specified by mapping the values to the animation-direction property.

Basic Syntax:

```
Unset
animation-direction:normal|reverse|alternate|
          alternate-reverse|initial|inherit;
```

Key Features of Values:

- **animation-direction:normal :-** It is default declaration, here animation moves the elements forward mode.
- **animation-direction:reverse :-** It is exactly vice versa of the "normal" and here the animation will be started in reverse direction.

- **animation-direction:alternate :-** In this property the animation is applied in an alternative manner and it will alternately switch between the direction.
- **animation-direction:alternate-reverse :-** In this property the animation is applied in an alternative manner and it will alternately switch between the direction, but here starting will be from reverse, then forward direction.
- **animation-direction:initial :-** It is used to set the property to its default value.
- **animation-direction:inherit :-** This declaration is used to inherit all the animation properties applicable from its parent.

Let us see the demonstration for all above properties with a specific example as stated below.

Example: Apply the most common animation-direction property to demonstrate their use cases.

HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="box ease">Ease</div>
    <div class="box linear">Linear</div>
    <div class="box ease-in">Ease In</div>
    <div class="box ease-out">Ease Out</div>
    <div class="box ease-in-out">Ease In Out</div>
</body>
</html>
```

CSS:

```
Unset
.box{
    width: 100px;
    height: 100px;
    margin:8px;
    color: azure;
    background: red;
    position: relative;
    justify-content: center;
    align-content: center;
    border-radius: 100px;
    animation-name: mymove;
    animation-duration:5s;
    text-align: center;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;
  }
  @keyframes mymove{
  from {left: 0px;}
  to {left: 200px;}
}


.ease {
  animation-timing-function: ease;
 background-color: blueviolet;
 animation-direction: normal;
}

.linear {
  animation-timing-function: linear;
  animation-direction:reverse;
}

.ease-in {
  animation-timing-function: ease-in;
  animation-direction: alternate;
}

.ease-out {
  animation-timing-function: ease-out;
  animation-direction: alternate-reverse;
}

.ease-in-out {
  animation-timing-function: ease-in-out;
  animation-direction:  initial;
}
```

OUTPUT:

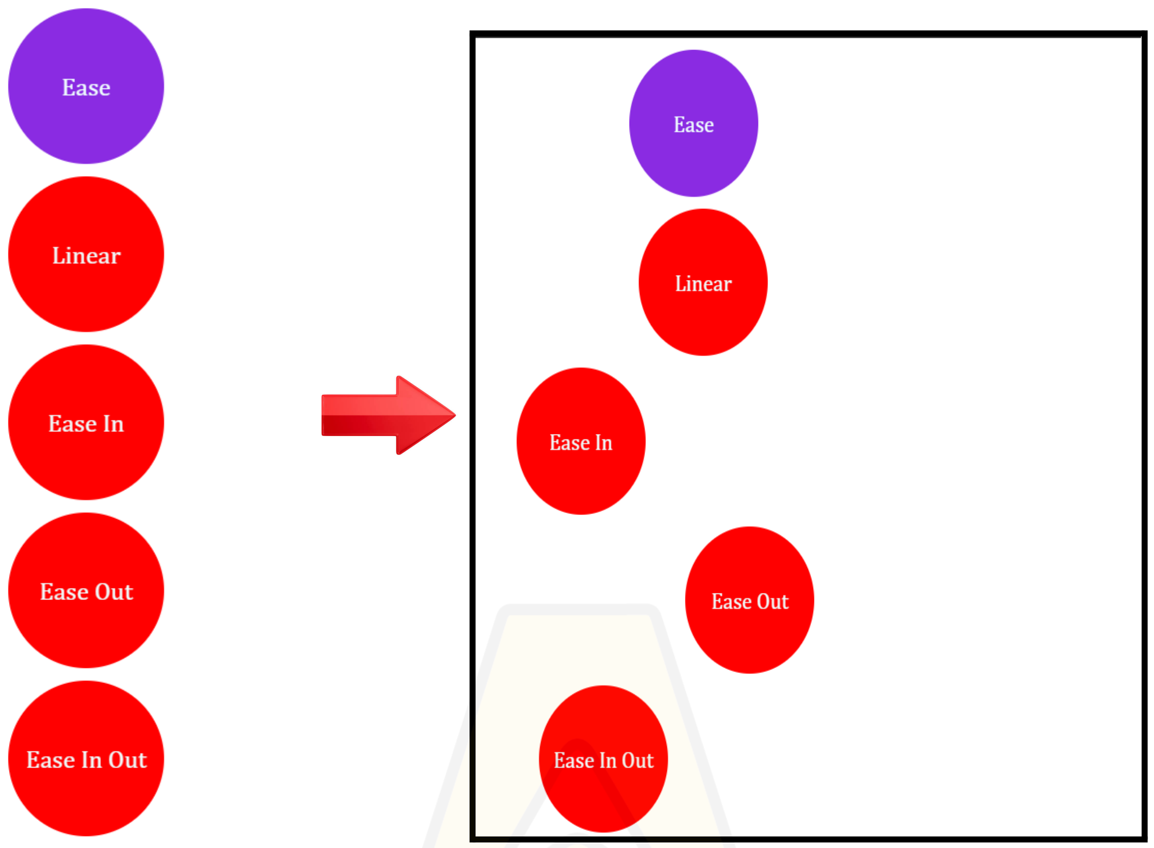


Fig: Initial Stage **Fig: During Execution**

1.2.7. animation-fill-mode: This property enables us to control how our animation is going to be applied on an element.

Note: A CSS animation does not affect the element before or after the first keyframe is played. This behaviour can be overridden by the animation-fill-mode property.

Example: Create a CSS animation and apply specific colour change after the execution of the animation, apply animation-fill-mode to reflect the colours to be swapped.

HTML:

```
Unset
<!DOCTYPE html>
```

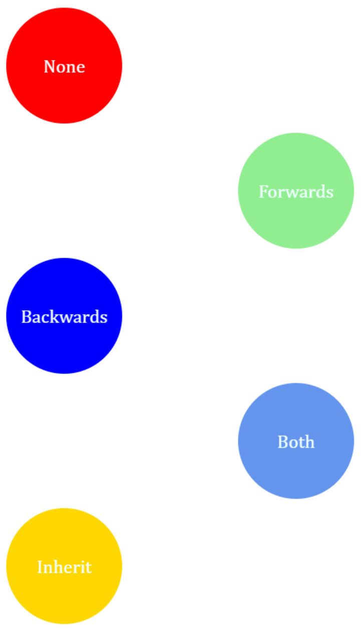
```
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="box none">None</div>
    <div class="box forwards">Forwards</div>
    <div class="box backwards">Backwards</div>
    <div class="box both">Both</div>
    <div class="box inherit">Inherit</div>
</body>
</html>
```

CSS:

```
Unset
.box{
    width: 100px;
    height: 100px;
    margin:8px;
    color: azure;
    background: red;
    position: relative;
    justify-content: center;
    align-content: center;
    border-radius: 100px;
    animation-name: mymove;
    animation-duration:5s;
    text-align: center;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;

  }
  @keyframes mymove{
  from {
    left: 0px;
    background-color: yellow;
  }
  to {
    left: 200px;

  }
}
```

```
.none{

  animation-fill-mode: none;
  background-color: red;
}

.forwards {
  animation-fill-mode: forwards;
  background-color: lightgreen;
}

.backwards {
  animation-fill-mode:backwards;
  background-color: blue;
}

 .both{

   animation-fill-mode: both;
   background-color: cornflowerblue;
}

.inherit{
  animation-fill-mode: inherit;
  background-color: gold;
}
```

OUTPUT:



1.2.8. animation-play-state:

Animation-play-state controls whether animations are playing or paused. A play state animation will run or a paused animation will halt.

Example: Create an animation and apply animation-play-state property to stop it on hovering the mouse.
HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="box ">1</div>
    <div class="box ">2</div>
    <div class="box ">3</div>

</body>
</html>
```

CSS:

```
Unset
.box{
    width: 100px;
    height: 100px;
    margin:8px;
    color: azure;
    background: red;
    position: relative;
    justify-content: center;
    align-content: center;
    border-radius: 100px;
    animation-name: mymove;
    animation-duration:5s;
    text-align: center;
    font-family: Cambria, Cochin, Georgia, Times, 'Times New Roman', serif;

  }
  @keyframes mymove{
  from {
    left: 0px;
    background-color: yellow;
  }
  to {
    left: 200px;

  }
}

.box:hover{
  animation-play-state:paused;
}
```

OUTPUT:

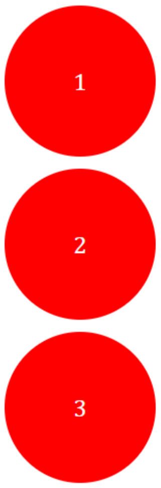


Fig: Initial Stage

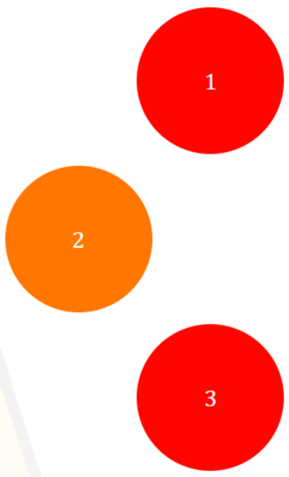


Fig: On keeping the mouse pointer at 2

2. CSS Webkit

-webkit is a vendor prefix used for properties that are supported only in WebKit-based browsers like Chrome, Safari, and newer versions of Opera. WebKit was the engine used by these browsers to render web pages.

Common -webkit properties used in CSS:

- **-webkit-appearance:** Controls the appearance of HTML elements.
- **-webkit-box-shadow:** Applies shadows to elements.
- **-webkit-transform:** Applies 2D or 3D transformations.
- **-webkit-transition:** Allows smooth transitions between states.

- **-webkit-text-stroke:** Adds stroke to text.

Example: Create a css stylesheet using CSS Webkit.

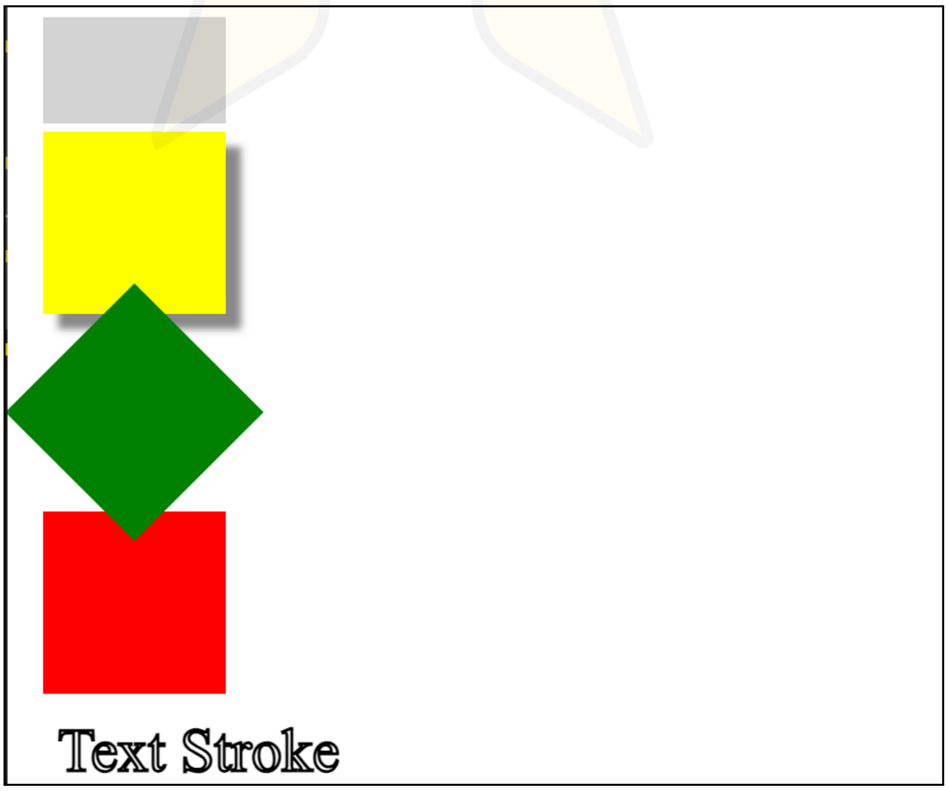
HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="appearance"></div>
    <div class="box-shadow"></div>
    <div class="transform"></div>
    <div class="transition"></div>
    <div class="text-stroke">Text Stroke</div>

</body>
</html>
```

CSS:

```
Unset
div{
  margin:5px 15px;
  padding:10px;
}
.appearance {
  -webkit-appearance: button;
  width: 100px;
  height: 50px;
  background-color: lightgrey;
}

.box-shadow {
  width: 100px;
  height: 100px;
  background-color: yellow;
  -webkit-box-shadow: 10px 10px 5px #888888;
}


.transform {
  width: 100px;
  height: 100px;
  background-color: green;
  -webkit-transform: rotate(45deg);
}


.transition {
  width: 100px;
  height: 100px;
  background-color: red;
  -webkit-transition: background-color 1s ease;
}

.transition:hover {
  background-color: blue;
}


.text-stroke {
  font-size: 40px;
  color: white;
  -webkit-text-stroke: 2px black;
}
```

OUTPUT:



3. CSS Custom Properties

CSS Custom properties are also known as CSS variables, are mainly to store values and reuse them throughout your CSS. This customization helps us to easily style and manage our CSS documents.

It mainly relies on two key declarations:-

1. " :root "
2. var(--<custom-value>)

3.1 ":root": defines custom properties for the entire document.

3.2 "var( )" : var( ) is used to access the value of the custom property.

Example: Implement use of css custom properties to apply background colour and colour to it.

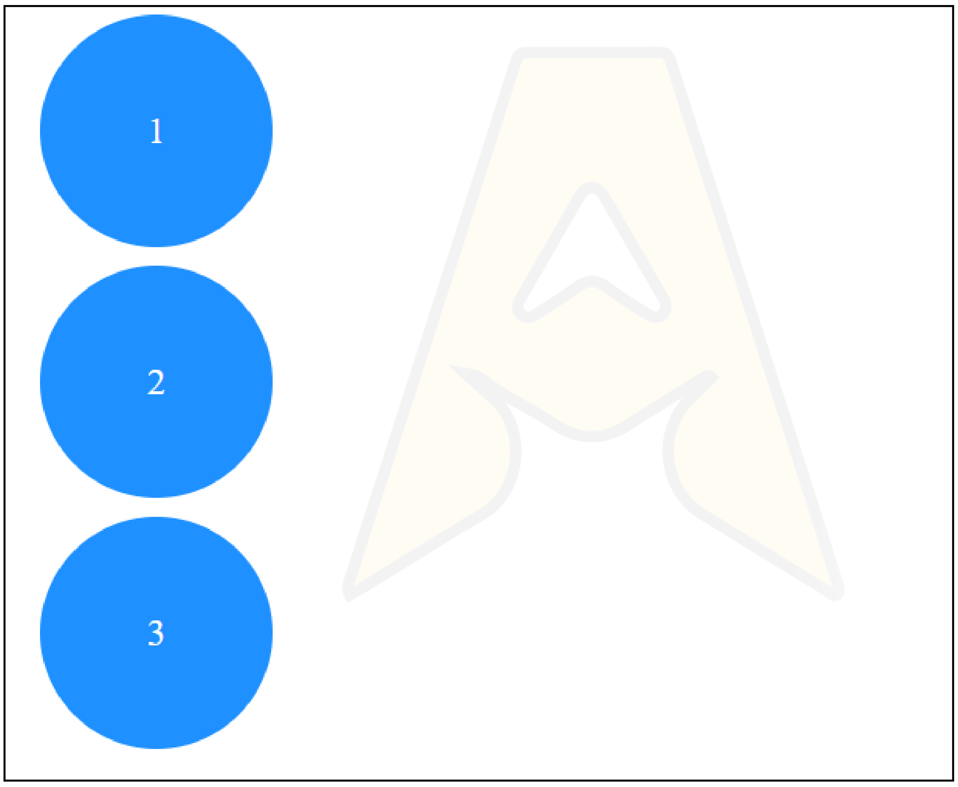
HTML:

```
Unset
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CSS Animation Timing Function</title>
    <link rel="stylesheet" href="animation.css">
</head>
<body>
    <div class="box ">1</div>
    <div class="box ">2</div>
    <div class="box ">3</div>

</body>
</html>
```

CSS:

```
Unset
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}

.box{
    width: 100px;
    height: 100px;
    margin:8px;
    color: var(--white);
    background: var(--blue);
    position: relative;
    justify-content: center;
    align-content: center;
    border-radius: 100px;
    text-align: center;
  }
```

OUTPUT:



THANK YOU

AIMERZ.ai
Aim.Act.Achieve