CSS Lists

Unordered Lists:

○ Coffee
○ Tea
○ Coca Cola

▪ Coffee
▪ Tea
▪ Coca Cola

Ordered Lists:

1. Coffee
2. Tea
3. Coca Cola

I. Coffee
II. Tea
III. Coca Cola

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

Different List Item Markers

The `list-style-type` property specifies the type of list item marker.

```
<!DOCTYPE html>

<html>

<head>

<style>

ul.a {
  list-style-type: circle;
}
ul.b {
  list-style-type: square;
}
ol.c {
  list-style-type: upper-roman;
}
ol.d {
  list-style-type: lower-alpha;
}
</style>
</head>
<body>
<h2>The list-style-type Property</h2>
<p>Example of unordered lists:</p>
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
<p>Example of ordered lists:</p>
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>

  <li>Coca Cola</li>

</ol>

<ol class="d">

  <li>Coffee</li>

  <li>Tea</li>

  <li>Coca Cola</li>

</ol>

</body>

</html>
```

An Image as The List Item Marker

The `list-style-image` property specifies an image as the list item marker:

```
<!DOCTYPE html>

<html>

<head>

<style>

ul {

  list-style-image: url('sqpurple.gif');

}

</style>

</head>

<body>

<h2>The list-style-image Property</h2>

<p>The list-style-image property specifies an image as the list item marker:</p>



<ul>

  <li>Coffee</li>

  <li>Tea</li>
  <li>Coca Cola</li>

</ul>

</body>

</html>
```

Remove Default Settings

The `list-style-type:none` property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add `margin:0` and `padding:0` to <ul> or <ol>:

Example

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

List - Shorthand property

The `list-style` property is a shorthand property. It is used to set all the list properties in one declaration:

Example

```
ul {
  list-style: square inside url("arrow.png");
}
```

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the <ol> or <ul> tag, affects the entire list, while properties added to the <li> tag will affect the individual list items:

Example

```
ol {
  background: #ff9999;
  padding: 20px;
}

ul {
  background: #3399ff;
  padding: 20px;
}

ol li {
  background: #ffe5e5;
  color: darkred;
  padding: 5px;
  margin-left: 35px;
}

ul li {
  background: #cce5ff;
  color: darkblue;
  margin: 5px;
}
```

Result:

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to

First Name	Last Name	Points	Points	Points	Points	Points	Points	Points	Points	Points	Points	Points	Points
Jill	Smith	50	50	50	50	50	50	50	50	50	50	50	50
Eve	Jackson	94	94	94	94	94	94	94	94	94	94	94	94
Adam	Johnson	67	67	67	67	67	67	67	67	67	67	67	67

display the full content:

Add a container element (like <div>) with `overflow-x:auto` around the <table> element to make it responsive:

Example

```
<div style="overflow-x:auto;">

<table>
... table content ...
</table>
```

```
</div>

<!DOCTYPE html>

<html>

<head>

<style>

table {

  border-collapse: collapse;

  width: 100%;

}


th, td {

  text-align: left;

  padding: 8px;

}


tr:nth-child(even) {background-color: #f2f2f2;}

</style>

</head>

<body>

<h2>Responsive Table</h2>

<p>A responsive table will display a horizontal scroll bar if the screen
is too

small to display the full content. Resize the browser window to see the
effect:</p>

<p>To create a responsive table, add a container element (like div) with
<strong>overflow-x:auto</strong> around the table element:</p>

<div style="overflow-x: auto;">

  <table>

    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
    </tr>
    <tr>
      <td>Jill</td>
      <td>Smith</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
      <td>50</td>
    </tr>
    <tr>
      <td>Eve</td>
```

```
      <td>Jackson</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
      <td>94</td>
    </tr>
    <tr>
      <td>Adam</td>
      <td>Johnson</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
      <td>67</td>
    </tr>
  </table>
</div>
```

```
</body>

</html>
```

Table Padding

To control the space between the border and the content in a table, use the `padding` property on <td> and <th> elements:

First Name	Last Name	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
th, td {
  padding: 15px;
  text-align: left;
}
```

Table Width and Height

The width and height of a table are defined by the `width` and `height` properties.

The example below sets the width of the table to 100%, and the height of the <th> elements to 70px:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
table {
  width: 100%;
}

th {
  height: 70px;
}
```

```
<!DOCTYPE html>

<html>

<head>

<style>

table, td, th {

  border: 1px solid black;

}

table {

  border-collapse: collapse;

  width: 100%;

}

th {

  height: 70px;

}

</style>

</head>

<body>

<h2>The width and height Properties</h2>

<p>Set the width of the table, and the height of the table header row:</p>

<table>

  <tr>

    <th>Firstname</th>

    <th>Lastname</th>

    <th>Savings</th>

  </tr>

  <tr>
    <td>Peter</td>

    <td>Griffin</td>

    <td>$100</td>

  </tr>

  <tr>

    <td>Lois</td>

    <td>Griffin</td>

    <td>$150</td>

  </tr>

  <tr>

    <td>Joe</td>

    <td>Swanson</td>

    <td>$300</td>

  </tr>

  <tr>

    <td>Cleveland</td>

    <td>Brown</td>

    <td>$250</td>

  </tr>

</table>

</body>

</html>
```

CSS Layout - The display Property

The `display` property is the most important CSS property for controlling layout.

The display Property

The `display` property specifies if/how an element is displayed.

Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is `block` or `inline`.

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The <div> element is a block-level element.

Examples of block-level elements:

- <div>
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline <span> element inside a paragraph.

Examples of inline elements:

- <span>
- <a>
- <img>

Display: none;

`display: none;` is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved.

The `<script>` element uses `display: none;` as default.

Override the Default Display Value

As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline `<li>` elements for horizontal menus:

```
<!DOCTYPE html>

<html>

<head>

<style>

li {

  display: inline;

}

</style>

</head>

<body>

<p>Display a list of links as a horizontal menu:</p>

<ul>

  <li><a href="/html/default.asp" target="_blank">HTML</a></li>

  <li><a href="/css/default.asp" target="_blank">CSS</a></li>

  <li><a href="/js/default.asp" target="_blank">JavaScript</a></li>

</ul>

</body>

</html>
```

The following example displays <span> elements as block elements:

```
<!DOCTYPE html>

<html>

<head>

<style>

span {
  display: block;

}

</style>

</head>

<body>

<h1>Display span elements as block elements</h1>

<span>A display property with</span> <span>a value of "block" results in</span> <span>a line break
between each span elements.</span>

</body>

</html>
```

The following example displays <a> elements as block elements:

```
<!DOCTYPE html>

<html>

<head>

<style>

a {

  display: block;

}

</style>

</head>

<body>

<h1>Display links as block elements</h1>

<a href="/html/default.asp" target="_blank">HTML</a>

<a href="/css/default.asp" target="_blank">CSS</a>

<a href="/js/default.asp" target="_blank">JavaScript</a>

</body>

</html>
```

Hide an Element - display:none or visibility:hidden?

Hiding an element can be done by setting the `display` property to `none`. The element will be hidden, and the page will be displayed as if the element is not there:

```
<!DOCTYPE html>

<html>

<head>

<style>

h1.hidden {

  display: none;

}

</style>

</head>

<body>

<h1>This is a visible heading</h1>

<h1 class="hidden">This is a hidden heading</h1>

<p>Notice that the h1 element with display: none; does not take up any space.</p>

</body>

</html>
```

However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

```
<!DOCTYPE html>

<html>

<head>

<style>

h1.hidden {

  visibility: hidden;

}

</style>

</head>

<body>



<h1>This is a visible heading</h1>
```

```
<h1 class="hidden">This is a hidden heading</h1>

<p>Notice that the hidden heading still takes up space.</p>

</body>

</html>
```

CSS Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element.

Transparent Image

The `opacity` property can take a value from 0.0 - 1.0. The lower the value, the more transparent:

```
<!DOCTYPE html>

<html>

<head>

<style>

img {

  opacity: 0.5;

}

</style>

</head>

<body>

<h1>Image Transparency</h1>

<p>The opacity property specifies the transparency of an element. The lower the value, the more
transparent:</p>

<p>Image with 50% opacity:</p>

<img src="img_forest.jpg" alt="Forest" width="170" height="100">

</body>

</html>
```

Transparent Hover Effect

The `opacity` property is often used together with the `:hover` selector to change the opacity on mouse-over:

```
<!DOCTYPE html>

<html>

<head>

<style>

img {

  opacity: 0.5;

}

img:hover {

  opacity: 1.0;

}

</style>

</head>

<body>

<h1>Image Transparency</h1>

<p>The opacity property is often used together with the :hover selector to change the opacity on mouse-over:</p>

<img src="img_forest.jpg" alt="Forest" width="170" height="100">

<img src="img_mountains.jpg" alt="Mountains" width="170" height="100">

<img src="img_5terre.jpg" alt="Italy" width="170" height="100">

</body>

</html>
```

Text in Transparent Box

```
<html>
<head>
<style>
div.background {
  background: url(forest.jpg) repeat;
  border: 2px solid black;
}

div.transbox {
  margin: 30px;
  background-color: #ffffff;
  border: 1px solid black;
  opacity: 0.6;
}

div.transbox p {
  margin: 5%;
  font-weight: bold;
  color: #000000;
}
</style>
</head>
<body>

<div class="background">
  <div class="transbox">
    <p>This is some text that is placed in the transparent box.</p>
  </div>
</div>

</body>
</html>
```

<u>CSS box-shadow Property</u>

```
<!DOCTYPE html>

<html>

<head>

<style>

#example1 {

  border: 1px solid;

  padding: 10px;

  box-shadow: 5px 10px;

}

#example2 {

  border: 1px solid;

  padding: 10px;

  box-shadow: 5px 10px #888888;

}
```

```
#example3 {
  border: 1px solid;
  padding: 10px;
  box-shadow: 5px 10px red;
}
</style>
</head>
<body>
<h1>The box-shadow Property</h1>
<p>The box-shadow property defines the shadow of an element:</p>
<h2>box-shadow: 5px 10px:</h2>
<div id="example1">
  <p>A div element with a shadow. The first value is the horizontal offset and the second value is the vertical offset. The shadow color will be inherited from the text color.</p>
</div>
<h2>box-shadow: 5px 10px #888888:</h2>
<div id="example2">
  <p>You can also define the color of the shadow. Here the shadow color is grey.</p>
</div>
<h2>box-shadow: 5px 10px red:</h2>
<div id="example3">
  <p>A red shadow.</p>
</div>
</body>
</html>
```

CSS Gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines three types of gradients:

- **Linear Gradients (goes down/up/left/right/diagonally)**
- **Radial Gradients (defined by their center)**
- **Conic Gradients (rotated around a center point)**

CSS Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

```
background-image: linear-gradient(direction, color-stop1, color-stop2,
...);
```

Direction - Top to Bottom (this is default)

The following example shows a linear gradient that starts at the top. It starts red, transitioning to yellow:

```
<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {

  height: 200px;

  background-color: red; /* For browsers that do not support gradients */

  background-image: linear-gradient(red, yellow);

}

</style>

</head>

<body>

<h1>Linear Gradient - Top to Bottom</h1>

<p>This linear gradient starts red at the top, transitioning to yellow at the bottom:</p>

<div id="grad1"></div>

</body>

</html>
```

Direction - Left to Right

The following example shows a linear gradient that starts from the left. It starts red, transitioning to yellow:

```
#grad {

 background-image: linear-gradient(to right, red , yellow);
}
```

Direction - Diagonal

You can make a gradient diagonally by specifying both the horizontal and vertical starting positions.

The following example shows a linear gradient that starts at top left (and goes to bottom right). It starts red, transitioning to yellow:

```
#grad {
  background-image: linear-gradient(to bottom right, red, yellow);
}
```

Rainbow Effects:

The following example shows how to create a linear gradient (from left to right) with the color of the rainbow and some text:

Example

```
#grad {
  background-image: linear-gradient(to right,
red,orange,yellow,green,blue,indigo,violet);
}
```

Repeating a linear-gradient

The repeating-linear-gradient () function is used to repeat linear gradients:

Example

A repeating linear gradient:

```
#grad {
  background-image: repeating-linear-gradient(red, yellow 10%, green 20%);
}
```

```
<!DOCTYPE html>

<html>

<head>

<style>

#grad1 {
  height: 200px;

  background-color: red; /* For browsers that do not support gradients */

  background-image: repeating-linear-gradient(red, yellow 10%, green 20%);

}


#grad2 {

  height: 200px;

  background-color: red; /* For browsers that do not support gradients */

  background-image: repeating-linear-gradient(45deg,red,yellow 7%,green
10%);

}


#grad3 {

  height: 200px;

  background-color: red; /* For browsers that do not support gradients */

  background-image: repeating-linear-gradient(190deg,red,yellow 7%,green
10%);

}


#grad4 {

  height: 200px;

  background-color: red; /* For browsers that do not support gradients */

  background-image: repeating-linear-gradient(90deg,red,yellow 7%,green
10%);

}


#grad5 {

  height: 200px;

  background-color: red; /* For browsers that do not support gradients */
  background-image: repeating-linear-gradient(45deg, red 0px, red 10px,
red 10px, yellow 10px, yellow 20px);

}

</style>

</head>

<body>

<h1>Repeating Linear Gradient</h1>

<div id="grad1"></div>

<p>A repeating gradient on 45deg axe starting red and finishing green:</p>

<div id="grad2"></div>

<p>A repeating gradient on 190deg axe starting red and finishing
green:</p>

<div id="grad3"></div>

<p>A repeating gradient on 90deg axe starting red and finishing green:</p>

<div id="grad4"></div>

<p>A repeating linear gradient with solid stripes:</p>

<div id="grad5"></div>

</body>

</html>
```

CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

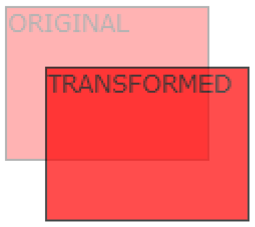
Mouse over the element below to see a 2D transformation:

- `transform`

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

- `translate()`
- `rotate()`
- `scaleX()`
- `scaleY()`
- `scale()`
- `skewX()`
- `skewY()`
- `skew()`
- `matrix()`

The translate() Method



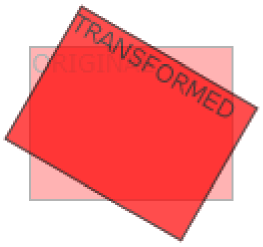
The `translate()` method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the <div> element 50 pixels to the right, and 100 pixels down from its current position:

Example

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

The rotate() Method



The `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:

Example

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

```
 <!DOCTYPE html>

<html>

<head>

<style>

div {

  width: 300px;

  height: 100px;

  background-color: yellow;
  border: 1px solid black;

}

div#myDiv {

  transform: rotate(20deg);

}

</style>

</head>

<body>

<h1>The rotate() Method</h1>

<p>The rotate() method rotates an element clockwise or counter-clockwise.</p>

<div>

This a normal div element.

</div>

<div id="myDiv">

This div element is rotated clockwise 20 degrees.

</div>



</body>

</html>
```

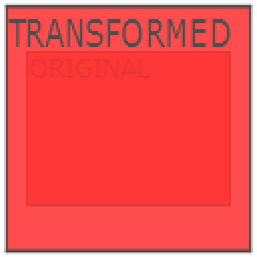
Using negative values will rotate the element counter-clockwise.

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

Example

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

The scale() Method



The `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:

Example

```
div {
  transform: scale(2, 3);
}
```

The following example decreases the <div> element to be half of its original width and height:

Example

```
div {
  transform: scale(0.5, 0.5);
}
```

The scaleX() Method

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

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

Example

```
div {
  transform: scaleX(2);
}
```

The following example decreases the <div> element to be half of its original width:

Example

```
div {
  transform: scaleX(0.5);
}
```

The scaleY() Method

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

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

Example

```
div {
  transform: scaleY(3);
}
```

The following example decreases the <div> element to be half of its original height:

```
div {
  transform: scaleY(0.5);
}
```

The skewX() Method

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

The following example skews the <div> element 20 degrees along the X-axis:

Example

```
div {
  transform: skewX(20deg);
}
```

The skewY() Method

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

The following example skews the <div> element 20 degrees along the Y-axis:

Example

```
div {
  transform: skewY(20deg);
}
```

The skew() Method

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

The following example skews the <div> element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

Example

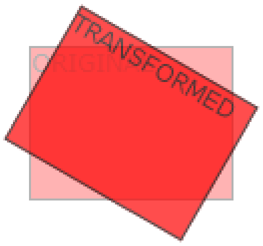
```
div {
  transform: skew(20deg, 10deg);
}
```

If the second parameter is not specified, it has a zero value. So, the following example skews the <div> element 20 degrees along the X-axis:

Example

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

The matrix() Method



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

The matrix() method take six parameters, containing mathematic functions, which allows you to rotate, scale, move (translate), and skew elements.

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

Example

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

```
<

!DOCTYPE html>

<html>

<head>

<style>

div {

  width: 300px;

  height: 100px;

  background-color: yellow;

  border: 1px solid black;

}

div#myDiv1 {

  transform: matrix(1, -0.3, 0, 1, 0, 0);

}

div#myDiv2 {

  transform: matrix(1, 0, 0.5, 1, 150, 0);

}

</style>

</head>
```

```
<body>

<h1>The matrix() Method</h1>

<p>The matrix() method combines all the 2D transform methods into one.</p>

<div>

This a normal div element.

</div>

<div id="myDiv1">

Using the matrix() method.

</div>

<div id="myDiv2">

Another use of the matrix() method.

</div>

</body>

</html>
```

CSS Animations

CSS allows animation of HTML elements without using JavaScript or Flash!

In this chapter you will learn about the following properties:

- `@keyframes`
- `animation-name`
- `animation-duration`
- `animation-delay`
- `animation-iteration-count`
- `animation-direction`
- `animation-timing-function`
- `animation-fill-mode`
- `animation`

The @keyframes Rule

When you specify CSS styles inside the `@keyframes` rule, the animation will gradually change from the current style to the new style at certain times.

To get an animation to work, you must bind the animation to an element.

The following example binds the "example" animation to the <div> element. The animation will last for 4 seconds, and it will gradually change the background-color of the <div> element from "red" to "yellow":

<!DOCTYPE html>

<html>

<head>

```
<style>

div {

  width: 100px;

  height: 100px;

  background-color: red;

  animation-name: example;

  animation-duration: 4s;

}

@keyframes example {

  from {background-color: red;}

  to {background-color: yellow;}

}

</style>

</head>

<body>

<h1>CSS Animation</h1>

<div></div>

<p><b>Note:</b> When an animation is finished, it goes back to its original style.</p>

</body>

</html>
```

In the example above we have specified when the style will change by using the keywords "from" and "to" (which represents 0% (start) and 100% (complete)).

It is also possible to use percent. By using percent, you can add as many style changes as you like.

The following example will change the background-color of the <div> element when the animation is 25% complete, 50% complete, and again when the animation is 100% complete:

```
<!DOCTYPE html>

<html>

<head>

<style>

div {

  width: 100px;

  height: 100px;

  background-color: red;
  animation-name: example;

  animation-duration: 4s;

}

@keyframes example {

  0%   {background-color: red;}

  25%  {background-color: yellow;}

  50%  {background-color: blue;}

  100% {background-color: green;}

}

</style>

</head>

<body>

<h1>CSS Animation</h1>

<div></div>

<p><b>Note:</b> When an animation is finished, it goes back to its original style.</p>

</body>

</html>
```

Run Animation in Reverse Direction or Alternate Cycles

The `animation-direction` property specifies whether an animation should be played forwards, backwards or in alternate cycles.

The animation-direction property can have the following values:

- `normal` - The animation is played as normal (forwards). This is default
- `reverse` - The animation is played in reverse direction (backwards)
- `alternate` - The animation is played forwards first, then backwards
- `alternate-reverse` - The animation is played backwards first, then forwards

The following example will run the animation in reverse direction (backwards):

```
<!DOCTYPE html>

<html>

<head>

<style>

div {
  width: 100px;

  height: 100px;

  background-color: red;

  position: relative;

  animation-name: example;

  animation-duration: 4s;

  animation-direction: reverse;

}

@keyframes example {

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

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

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

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

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

}

</style>

</head>

<body>

<h1>CSS Animation</h1>

<p>The animation-direction property specifies whether an animation should be played forwards, backwards or in
alternate cycles. The following example will run the animation in reverse direction (backwards):</p>

<div></div>

</body> </html>
```

Assignments :-

1. Create a list using csss list-style type.
2. create table using border,size, alignment,style.
3.create a page using position - attribute.
4. Define layout includes- Header,content , foooter
5. Create a Section (div) and apply animation - change the color Blue to Red.