Unit-5

<meta> tag(setting viewport for responsive design)

The viewport is the user's visible area of a web page. It varies with the device - it will be smaller on a mobile phone than on a computer screen.

You should include the following `<meta>` element in all your web pages:

```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

This gives the browser instructions on how to control the page's dimensions and scaling.

The `width=device-width` part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The `initial-scale=1.0` part sets the initial zoom level when the page is first loaded by the browser.

```
<meta name="viewport"
      content="width=device-width,
               initial-scale=1.0,
               user-scalable=no,
               maximum-scale=0.5,
               minimum-scale=0.1"/>
```

`minimum-scale`[0.0-10.0] gives minimum size to allow to zoom-out.

`Maximum-scale`[0.0-10.0] gives maximum size to allow to zoom-in

CSS Media queries

The `@media` rule is used in media queries to apply different styles for different media types/devices.

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet (responsive web design) to desktops, laptops, tablets, and mobile phones.

Syntax

```
@media not|only mediatype and (mediafeature and|or|not mediafeature)
{
  CSS-Code;
}
```

meaning of the **not**, **only** and **and** keywords:

not: The not keyword inverts the meaning of an entire media query.

only: The only keyword prevents older browsers that do not support media queries with media features from applying the specified styles. **It has no effect on modern browsers.**

and: The and keyword combines a media feature with a media type or other media features. (mediatype: print, screen, or speech).

They are all optional. However, if you use **not** or **only**, you must also specify a media type.

EXAMPLE:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
body {
  background-color: lightblue;
}
@media screen and (min-width: 400px) {
  body {
    background-color: lightgreen;
  }
}
@media screen and (min-width: 800px) {
  body {
    background-color: lavender;
  }
}
</style>
</head>
<body>

<h1>Resize the browser window to see the effect!</h1>
```

```
<p>Use mediaqueries to set the background-color to lavender if the viewport is 800 pixels wide or wider, to lightgreen if the viewport is between 400 and 799 pixels wide. If the viewport is smaller than 400 pixels, the background-color is lightblue.</p>

</body>
</html>
```

Media queries can also be used to change layout of a page depending on the orientation of the browser. You can have a set of CSS properties that will only apply when the browser window is wider than its height, a so called "Landscape" orientation.

EXAMPLE:

```
<html>
<head>
<meta name="viewport" content="width=device-width, initial-scale=1.0">
<style>
body {
  background-color: lightgreen;
}

@media not screen and (orientation: landscape) {
  body {
    background-color: lightblue;
  }
}
</style>
</head>
<body>

<p>Resize the browser window. When the width of this document is larger than the height, the background-color is "lightblue", otherwise it is "lightgreen".</p>

</body>
</html>
```

CSS style images

The styling of an image in CSS is similar to the styling of an element by using the borders and margins. There are multiple CSS properties such as **border** property, **height** property, **width** property, etc. that helps us to style an image.

Thumbnail Image

The border property is used to make a thumbnail image.

EXAMPLE

```
<html>
<head>
   <style>
img{
border: 2px solid red;
border-radius:5px;
padding:10px;
}
h2{
color:red;
}
   </style>
</head>

<body>
 <h1>Thumbnail Image</h1>
<img src="jtp.png"></img>
<h2>Welcome to javaTpoint</h2>
</body>
</html>
```

Transparent image

To make an image transparent, we have to use the **opacity** property. The value of this property lies between 0.0 to 1.0, respectively.

EXAMPLE

```
<html>
<head>
   <style>
img{
border: 2px solid red;
border-radius:5px;
padding:10px;
opacity:0.3;
}
h2{
```

```
color:red;
}
   </style>
</head>

<body>
 <h1>Transparent Image</h1>
<img src="jtp.png"></img>
<h2>Welcome to javaTpoint</h2>
</body>
</html>
```

Rounded image

The **border-radius** property sets the radius of the bordered image. It is used to create the rounded images. The possible values for the rounded corners are given as follows:

- **border-radius:** It sets all of the four border-radius property.
- **border-top-right-radius:** It sets the border of the top-right corner.
- **border-top-left-radius:** It sets the border of the top-left corner.
- **border-bottom-right-radius:** It sets the border of the bottom-right corner.
- **border-bottom-left-radius:** It sets the border of the bottom-left corner.

Example

```
 <!DOCTYPE html>
<html>
<head>
   <style>
#img1{
border: 2px solid green;
border-radius:10px;
padding:5px;
}
#img2{
border: 2px solid green;
border-radius:50%;
padding:5px;
}

h2{
color:red;
```

```
}
   </style>
</head>

<body>
 <h1>Rounded Image</h1>
<img src="jtp.png" id="img1"></img>
<h2>Welcome to javaTpoint</h2>

 <h1>Circle Image</h1>
<img src="jtp.png" id="img2"></img>
<h2>Welcome to javaTpoint</h2>
</body>
</html>
```

Responsive Image

It automatically adjusts to fit on the screen size. It is used to adjust the image to the specified box automatically.

Example

```
<html>
<head>
<style>
#img1{
max-width:auto;
height:auto;
}
h2{
color:red;
}
   </style>
</head>

<body>
 <h1>Responsive image</h1>
 <h2>You can resize the browser to see the effect</h2>
<img src="jtp.png" id="img1" width="1000" height="300"></img>
<h2>Welcome to javaTpoint</h2>
</body>
</html>
```

Program: Write down HTML/CSS code that add some text to an image in the bottom right corner and top left corner with use of opacity property of image.

```
<html>
<head>
<style>
.container {
  position: relative;
}

.topleft {
  position: absolute;
  top: 8px;
  left: 16px;
  font-size: 18px;
}
.bottomright {
  position: absolute;
  bottom: 8px;
  right: 16px;
  font-size: 18px;
}



</style>
</head>
<body>

<h2>Image Text</h2>

<p>Add some text to an image in the top left corner:</p>

<div class="container">
  <img src="C:\Users\hiral\Downloads\download.jpg" alt="Cinque Terre"
style="width:100%; height:300;opacity:0.3;">
  <div class="topleft">Top Left</div>
  <div class="bottomright">Bottom Right</div>
</div>

</body>
</html>
```

CSS 2D Transforms

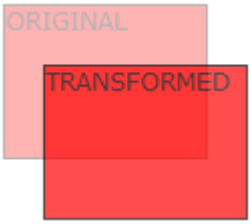
CSS transforms allow you to move, rotate, scale, and skew elements.

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

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

- `translate()`
- `rotate()`
- `scale()`
- `skew()`

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 element 50 pixels to the right, and 100 pixels down from its current position:

EXAMPLE

```
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
  }
div:hover
{ transform: translate(50px,100px);
}
</style>
</head>
<body>

<h1>The translate() Method</h1>
<p>The translate() method moves an element from its current position:</p>

<div>
This div element is moved 50 pixels to the right, and 100 pixels down from its current position.
</div>

</body>
```

```
</html>
```

The rotate() Method

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

For counter-clockwise give value in (-)negative.

The following example rotates the element clockwise with 20 degrees:

EXAMPLE

```
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}

div:hover
{  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>
```

```
</body>
</html>
```

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 element to be two times of its original width, and three times of its original height:

EXAMPLE

```
<html>
<head>
<style>
div {
  margin: 150px;
  width: 200px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}
div:hover
{  transform: scale(2,3);
}
</style>
</head>
<body>

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

<p>The scale() method increases or decreases the size of an element.</p>

<div>
This div element is two times of its original width, and three times of its original height.
</div>

</body>
</html>
```

The scaleX() Method

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

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

Example

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

    }
```

The following example decreases the 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 element to be three times of its original height:

Example

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

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

Example

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

The skew() Method

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

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

Example

```
<html>
<head>
<style>
div {
  width: 300px;
  height: 100px;
  background-color: yellow;
  border: 1px solid black;
}

div:hover {
  transform: skew(20deg,10deg);
}
</style>
</head>
<body>

<h1>The skew() Method</h1>
<p>The skew() method skews an element into a given angle.</p>

<div>
This a normal div element.
</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.

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:

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

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);
}
```

EXAMPLE

```
<html>

<head>

<style>

#grad1 {

  height: 200px;


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

}

</style>

</head>

<body>


<h1>Linear Gradient - Left to Right</h1>

<p>This linear gradient starts red at the left, transitioning to yellow (to the right):</p>


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

```
</body>

</html>
```

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);
}
```

Using Multiple Color Stops

The following example shows a linear gradient (from top to bottom) with multiple color stops:

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

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

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

EXAMPLE

```
<html>

<head>

<style>

#grad1 {

  height: 55px;

  background-image: linear-gradient(to right, red, orange, yellow,
    green, blue, indigo, violet);

}

</style>
```

```
</head>

<body>


<div id="grad1" style="text-
    align:center;margin:auto;color:#888888;font-size:40px;font-
    weight:bold">

Rainbow Background

</div>


</body>

</html>
```

Repeating a linear-gradient

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

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

EXAMPLE

```
<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 Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

Syntax

```
background-image: radial-gradient(shape size at position, start-color, ..., last-color);
```

By default, shape is ellipse, size is farthest-corner, and position is center.

Radial Gradient - Evenly Spaced Color Stops (this is default)

The following example shows a radial gradient with evenly spaced color stops:

```
#grad {
  background-image: radial-gradient(red, yellow, green);
}
```

Radial Gradient - Differently Spaced Color Stops

The following example shows a radial gradient with differently spaced color stops:

```
#grad {
  background-image: radial-gradient(red 5%, yellow 15%, green 60%);
}
```

Set Shape

The shape parameter defines the shape. It can take the value circle or ellipse. The default value is ellipse.

The following example shows a radial gradient with the shape of a circle:

```
#grad {
  background-image: radial-gradient(circle, red, yellow, green);
}
```

Repeating a radial-gradient

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

```
#grad {
  background-image: repeating-radial-gradient(red, yellow 10%, green 15%);
}
```

EXAMPLE

```
<html>

<head>

<style>

#grad1 {

  height: 150px;

  width: 200px;

  background-image: repeating-radial-gradient(red, yellow 10%, green
15%);

}

#grad2 {

  height: 150px;

  width: 200px;

  background-image: radial-gradient(red 5%, yellow 15%, green 60%);



}

</style>

</head>

<body>

<h1>Repeating Radial Gradient</h1>

<div id="grad1"></div>
<div id="grad2"></div>

</body>

</html>
```

CSS Conic Gradients

A conic gradient is a gradient with color transitions rotated around a center point.

To create a conic gradient you must define at least two colors.

Syntax

```
background-image: conic-gradient([from angle]
[at position,] color [degree], color [degree], ...);
```

By default, *angle* is 0 deg and *position* is center.

If no *degree* is specified, the colors will be spread equally around the center point.

Conic Gradient: Three Colors

The following example shows a conic gradient with three colors:

```
#grad {
  background-image: conic-gradient(red, yellow, green);
}
```

Conic Gradient: Colors and Degrees

The following example shows a conic gradient with three colors and a degree for each color:

```
#grad {
  background-image: conic-gradient(red 45deg, yellow 90deg, green
210deg);
}
```

Repeating a Conic Gradient

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

```
#grad {
  background-image: repeating-conic-gradient(red 10%, yellow 20%);
  border-radius: 50%;

}
```

Create Pie Charts

Just add `border-radius: 50%` to make the conic gradient look like a pie:

```
#grad {
  background-image: conic-gradient(red, yellow, green, blue, black);
  border-radius: 50%;


}
```

CSS Transitions

CSS transitions allows you to change property values smoothly, over a given duration.

properties:

- `transition`
- `transition-delay`
- `transition-duration`
- `transition-property`

How to Use CSS Transitions?

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

The `transition-delay` property specifies a delay (in seconds) for the transition effect.

The following example has a 1 second delay before starting:

```
div {
  transition-delay: 1s;
}
```

The following example shows a 100px * 100px red element. The element has also specified a transition effect for the width property, with a duration of 2 seconds:

Example

<html>
<head>
<style>
div {

```
  width: 100px;
  height: 100px;
  background: red;
  transition-property: width;
  transition-duration: 2s;
  transition-delay: 1s;
}

div:hover {
  width: 300px;
}
</style>
</head>
<body>

<h1>The transition Properties Specified One by One</h1>

<p>Hover over the div element below, to see the transition effect:</p>

<div></div>

<p><b>Note:</b> The transition effect has a 1 second delay before
starting.</p>

</body>
</html>
```

Example

Write HTML and CSS script to design one login form having fields like username,
password,submit and reset button.while hovering each control, it should be expanded
around 20% horizontally and vertically with delay of 2 second.expansion should take place
in 4 seconds.

SOLUTION:-

```
<html>
<head>
<style>
.sub
{width:10%;height:10%;
 transition:height,width;
```

```
 transition-delay:2s;
 transition-duration:4s;
}
.sub:hover{
width:5%;height:5%;
 }
input{
 width:10%;height:10%;
 transition:height,width;
 transition-delay:2s;
 transition-duration:4s;
 }
 input:hover{
 width:20%;height:20%;
 }
</style>
</head>
<body>
<form>
<label>Username:
<input type="text" name="uname"/></label><br>
<label>Password:
<input type="password" name="upassword"/></label><br><br>

<input type="submit" class="sub" name="sbtn" value="Submit"/>
<input type="reset" class="sub" name="rbtn" value="Reset"/>
</form>
</body>
</html>
```

CSS Animations

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

CSS Animation Properties

The following table lists the @keyframes rule and all the CSS animation properties:

Property	Description

@keyframes	Specifies the animation code
animation	A shorthand property for setting all the animation properties
animation-delay	Specifies a delay for the start of an animation
animation-direction	Specifies whether an animation should be played forwards, backwards or in alternate cycles
animation-duration	Specifies how long time an animation should take to complete one cycle
animation-name	Specifies the name of the @keyframes animation
animation-iteration-count	Specifies the number of times an animation should be played

What are CSS Animations?

An animation lets an element gradually change from one style to another.

You can change as many CSS properties you want, as many times as you want.

To use CSS animation, you must first specify some keyframes for the animation.

Keyframes hold what styles the element will have at certain times.

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 animation code */
@keyframes example {
  from {background-color: red;}
  to {background-color: yellow;}
}

/* The element to apply the animation to */
div {
  width: 100px;
  height: 100px;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
}
```

Note: The `animation-duration` property defines how long an animation should take to complete. If the `animation-duration` property is not specified, no animation will occur, because the default value is 0s (0 seconds).

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

Example

```
<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>
</body>
</html>
```

Example

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

```
<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>

</body>
</html>
```

Example

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

```
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
}

@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>
<div></div>

</body>
</html>
```

Delay an Animation

The `animation-delay` property specifies a delay for the start of an animation.The following example has a 2 seconds delay before starting the animation:

Example

```
div {
  width: 100px;
```

```
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: 2s;
}
```

Negative values are also allowed. If using negative values, the animation will start as if it had already been playing for *N* seconds.

In the following example, the animation will start as if it had already been playing for 2 seconds:

Example

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: -2s;
}
```

Set How Many Times an Animation Should Run

The `animation-iteration-count` property specifies the number of times an animation should run.

The following example will run the animation 3 times before it stops:

Example

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: 3;
}
```

The following example uses the value "infinite" to make the animation continue for ever:

Example

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: infinite;
}
```

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):

Example

```
div {
  width: 100px;
  height: 100px;
  position: relative;
  background-color: red;
  animation-name: example;
  animation-duration: 4s;
  animation-direction: reverse;
}
```

CSS **display** Property

The `display` property specifies the display behavior (the type of rendering box) of an element.

Syntax

```
display: value;
```

Property Values

Value	Description
inline	Displays an element as an inline element (like). Any height and width properties will have no effect
block	Displays an element as a block element (like <p>). It starts on a new line, and takes up the whole width
inline-block	Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values
none	The element is completely removed

Example

```
<html>
<head>
<style>
p {color: red;}

p.ex1 {display: none;}
p.ex2 {display: inline;}
p.ex3 {display: block;}
p.ex4 {display: inline-block;}
</style>
</head>
<body>
<h1>The display Property</h1>
```

```
<h2>display: none:</h2>
<div>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper
diam at erat pulvinar, at pulvinar felis blandit. <p class="ex1">HELLO
WORLD!</p> Vestibulum volutpat tellus diam, consequat gravida libero
rhoncus ut.
</div>

<h2>display: inline:</h2>
<div>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper
diam at erat pulvinar, at pulvinar felis blandit. <p class="ex2">HELLO
WORLD!</p> Vestibulum volutpat tellus diam, consequat gravida libero
rhoncus ut.
</div>

<h2>display: block:</h2>
<div>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper
diam at erat pulvinar, at pulvinar felis blandit. <p class="ex3">HELLO
WORLD!</p> Vestibulum volutpat tellus diam, consequat gravida libero
rhoncus ut.
</div>

<h2>display: inline-block:</h2>
<div>
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper
diam at erat pulvinar, at pulvinar felis blandit. <p class="ex4">HELLO
WORLD!</p> Vestibulum volutpat tellus diam, consequat gravida libero
rhoncus ut.
</div>

</body>
</html>
```

CSS **Tooltip**

A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element:

Create a tooltip that appears when the user moves the mouse over an element:

Example

```
<html>
<style>
.tooltip {
```

```
  position: relative;
  display: inline-block;
  border-bottom: 1px dotted black;
}

.tooltip .tooltiptext {
  visibility: hidden;
  width: 120px;
  background-color: black;
  color: #fff;
  text-align: center;
  border-radius: 6px;
  padding: 5px 0;

  /* Position the tooltip */
  position: absolute;

}

.tooltip:hover .tooltiptext {
  visibility: visible;
}
</style>
<body style="text-align:center;">

<h2>Basic Tooltip</h2>

<p>Move the mouse over the text below:</p>

<div class="tooltip">Hover over me
  <span class="tooltiptext">Tooltip text</span>
</div>

<p>Note that the position of the tooltip text isn't very good. Go back
to the tutorial and continue reading on how to position the tooltip in
a desirable way.</p>

</body>
</html>
```

Positioning Tooltips

Right Tooltip

```
.tooltip .tooltiptext {
  top: -5px;
  left: 105%;
}
```

Left Tooltip

```
.tooltip .tooltiptext {
  top: -5px;
  right: 105%;
}
```

Top Tooltip

```
.tooltip .tooltiptext {
  bottom: 100%;
  left: 0%;
}
```

Bottom Tooltip

```
.tooltip .tooltiptext
{
  top: 100%;
  left: 0%;
  }
```

Flip an Image

to flip an image (add a mirror effect) with CSS.

Example

```
<html>
<head>
<style>
img:hover {

  transform:scalex(-1);
}
</style>
</head>
<body>

<h2>Flip an Image</h2>
<p>Move your mouse over the image.</p>

<img src="img_paris.jpg" alt="Paris" width="400" height="300">

</body>
</html>
```

CSS **Buttons**

we use the button tag to create a button, but by using CSS properties, we can style the buttons.
There are multiple properties available that are used to style the button element.

background-color

this property is used for setting the background color of the button element.

Syntax

```
element {
   // background-color style
}
```

border

It is used to set the border of the button. It is the shorthand property for border-width, border-color, and border-style.

Syntax

```
element {
   // border style
}
```

border-radius

It is used to make the rounded corners of the button. It sets the border radius of the button.

Syntax

```
element {
   // border-radius property
}
```

Example

```
<html>
<head>
<style>
```

```
.button {
  background-color: red;
  border-radius: 50%;
  border: 2px solid green;
  color: white;
  padding: 15px 32px;
  text-align: center;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  cursor: pointer;
}
</style>
</head>
<body>

<h2>CSS Buttons</h2>

<button>Default Button</button>
<a href="#" class="button">Link Button</a>
<button class="button">Button</button>
<input type="button" class="button" value="Input Button"/>

</body>
</html>
```

You can use different cursor Properties

The `cursor` property specifies the mouse cursor to be displayed when pointing over an element.

Value	Description
alias	The cursor indicates an alias of something is to be created
auto	Default. The browser sets a cursor

grab	The cursor indicates that something can be grabbed
not-allowed	The cursor indicates that the requested action will not be executed
pointer	The cursor is a pointer and indicates a link
progress	The cursor indicates that the program is busy (in progress)
wait	The cursor indicates that the program is busy
zoom-in	The cursor indicates that something can be zoomed in
zoom-out	The cursor indicates that something can be zoomed out

Hoverable Buttons

Use the `:hover` selector to change the style of a button when you move the mouse over it.
Use the `transition-duration` property to determine the speed of the "hover" effect:

Example

```
<!DOCTYPE html>
<html>
<head>
<style>
```

```
.button {
  background-color:white;
  border:1px solid red;
  color: black;
  padding: 16px 32px;
  text-align: center;
  text-decoration: none;
  display: inline-block;
  font-size: 16px;
  margin: 4px 2px;
  transition-duration: 0.4s;
  cursor: pointer;
}

.button:hover {
  background-color: #4CAF50;
  color: white;
}
</style>
</head>
<body>

<h2>Hoverable Buttons</h2>
<button class="button">Green</button>
</body>
</html>
```

CSS Pagination

Simple Pagination

If you have a website with lots of pages, you may wish to add some sort of pagination to each page:

« 1 2 3 4 5 6 »

Example

```
<html>
<head>
<style>
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
```

```
  float: left;
  padding: 8px 16px;
  text-decoration: none;
}
</style>
</head>
<body>

<h2>Simple Pagination</h2>

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a href="#">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>

</body>
</html>
```

Active and Hoverable Pagination

« 1 2 3 4 5 6 7 »

Highlight the current page with an `.active` class, and use the `:hover` selector to change the color of each page link when moving the mouse over them:

Example

```
<html>
<head>
<style>
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
  float: left;
  padding: 8px 16px;
  text-decoration: none;
```

```
}

.pagination .active {
  background-color: #4CAF50;
  color: white;
}

.pagination a:hover:not(.active) {background-color: #ddd;}
</style>
</head>
<body>

<h2>Active and Hoverable Pagination</h2>

<p>Move the mouse over the numbers.</p>

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a class="active" href="#">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>

</body>
</html>
```

Rounded Active and Hoverable Buttons

Add the `border-radius` property if you want a rounded "active" and "hover" button:

Example

```
<html>
<head>
<style>
.pagination {
  display: inline-block;
}

.pagination  a {
  color: black;
```

```
  float: left;
  padding: 8px 16px;
  text-decoration: none;
}

.pagination .active {
  background-color: #4CAF50;
  color: white;
  border-radius: 5px;
}

.pagination a:hover:not(.active) {
  background-color: #ddd;
  border-radius: 5px;
}
</style>
</head>
<body>

<h2>Rounded Active and Hover Buttons</h2>

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a href="#" class="active">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>

</body>
</html>
```

Bordered Pagination

«	1	2	3	4	5	6	7	»

Use the `border` property to add borders to the pagination:

Example

```
<html>
<head>
<style>
```

```
.pagination {
  display: inline-block;
}

.pagination a {
  color: black;
  float: left;
  padding: 8px 16px;
  text-decoration: none;
  transition: background-color .3s;
  border: 1px solid #ddd;
}

.pagination .active {
  background-color: #4CAF50;
  color: white;
  border: 1px solid #4CAF50;
}

.pagination a:hover:not(.active) {background-color: #ddd;}
</style>
</head>
<body>

<h2>Pagination with Borders</h2>

<div class="pagination">
  <a href="#">&laquo;</a>
  <a href="#">1</a>
  <a href="#" class="active">2</a>
  <a href="#">3</a>
  <a href="#">4</a>
  <a href="#">5</a>
  <a href="#">6</a>
  <a href="#">&raquo;</a>
</div>

</body>
</html>
```

CSS Multiple Columns

The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers:

Multi-column Properties

- column-count
- column-gap
- column-rule-style
- column-rule-width
- column-rule-color

Property	Description
column-count	Specifies the number of columns an element should be divided into
column-gap	Specifies the gap between the columns
column-rule-color	Specifies the color of the rule between columns
column-rule-style	Specifies the style of the rule between columns (solid,none,dotted,dashed)
column-rule-width	Specifies the width of the rule between columns

Example

```
<html>
<head>
<style>
.newspaper {
  column-count: 3;
  column-gap: 40px;
  column-rule-style: solid;
  column-rule-width: 1px;
  column-rule-color: lightblue;
}
</style>
</head>
<body>
```

```
<h1>Set the Rule Color</h1>

<div class="newspaper">
Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.
</div>

</body>
</html>
```

CSS var() Function

The var() function is used to insert the value of a CSS variable.
The variables store the values and have a scope in which they can be used.

CSS variables are advantageous because they allow us to reuse the same value at multiple places. The name of the variable is easy to understand and use, as compared to the color values.

Syntax:

```
element {
  --main-color: brown;
}
```

A variable in CSS is defined by using the two dashes (--) at the beginning, followed by the name, which is case-sensitive.

In the above syntax, the *element* indicates the selector that specifies the scope of the custom property. If we define the custom properties on the **:root** pseudo-class, then it will be globally applied to our HTML document. The names of the custom properties are case-sensitive

Example

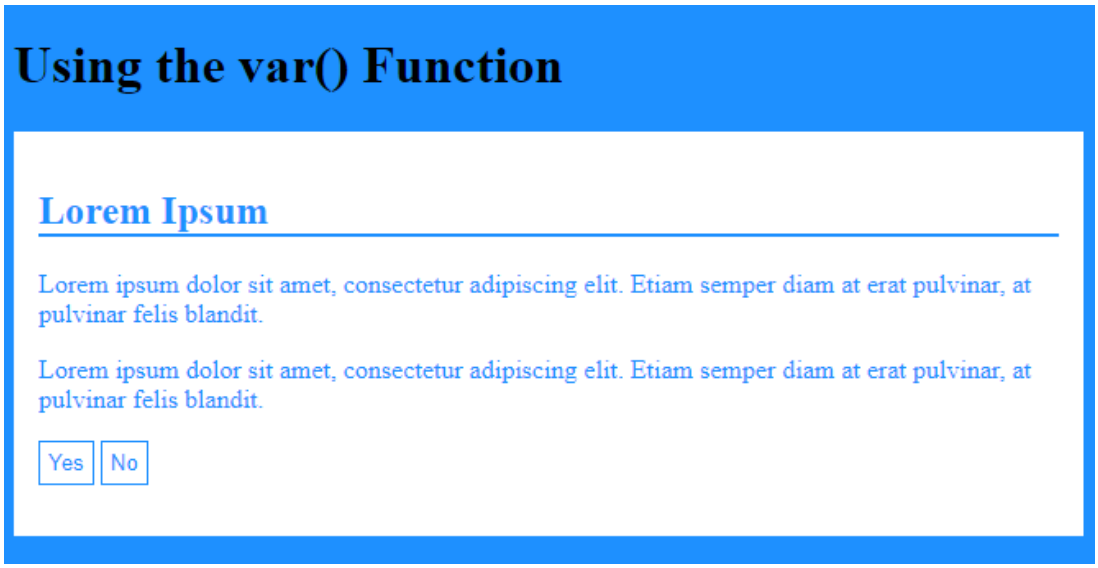
```
<html>
<head>
<style>
```

```
 :root
  { --blue:#1234EE;
 --white:#FFFFFF;
}

 h1
  { --blue:113355;
    Color:var(--blue);
  }
P
{ color:var(--white);
  Background-color:var(--blue);
}
</style>
</head>
<body>
<h1>this is an example of local and globle variable</h1>
<p> The var() function in CSS is used to insert the custom property value. </p>
</body>
</html>
```

Example

Write a code to display following output:



Solution:

```
<html>
<head>
<style>
:root {
  --blue: #1e90ff;
```

```
  --white: #ffffff;
}

body {
  background-color: var(--blue);
}

h2 {
  border-bottom: 2px solid var(--blue);
}

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
}

button {
  background-color: var(--white);
  color: var(--blue);
  border: 1px solid var(--blue);
  padding: 5px;
}
</style>
</head>
<body>

<h1>Using the var() Function</h1>

<div class="container">
  <h2>Lorem Ipsum</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.</p>
  <p>
    <button>Yes</button>
    <button>No</button>
  </p>
</div>

</body>
</html>
```