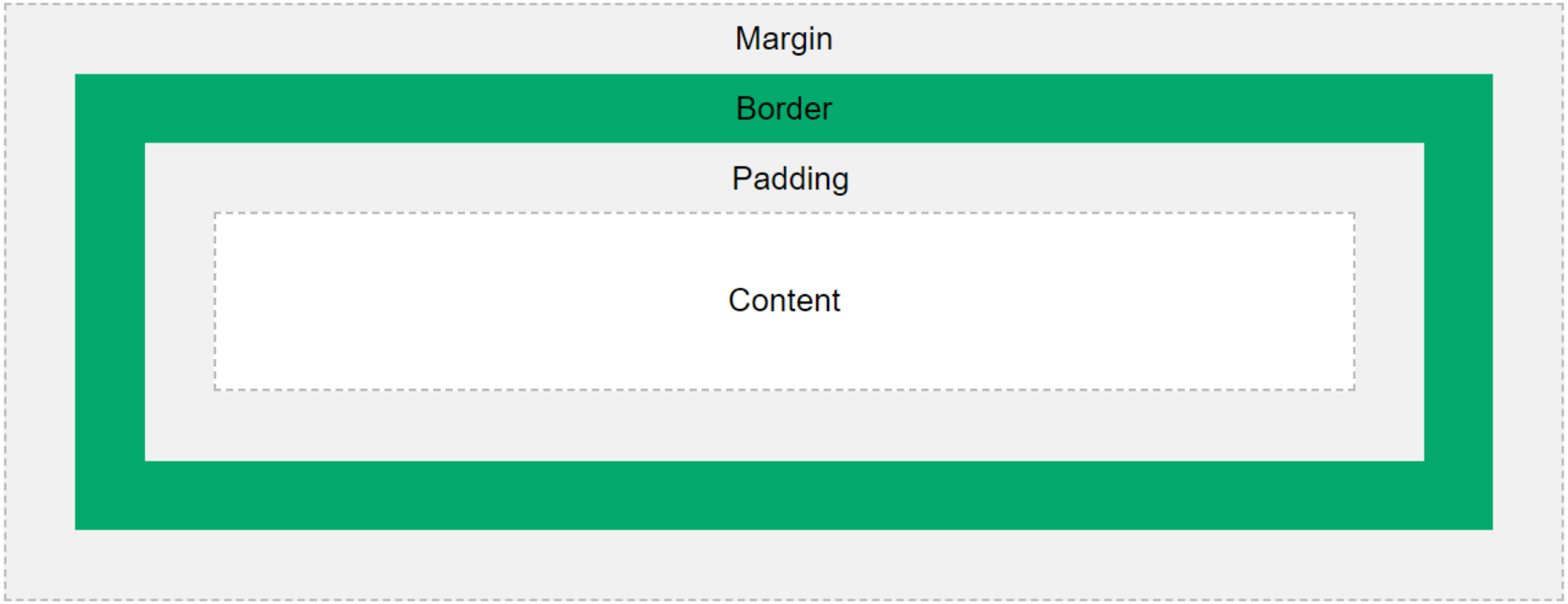
Cascading Style Sheet

UNIT 3: CSS

Part-2

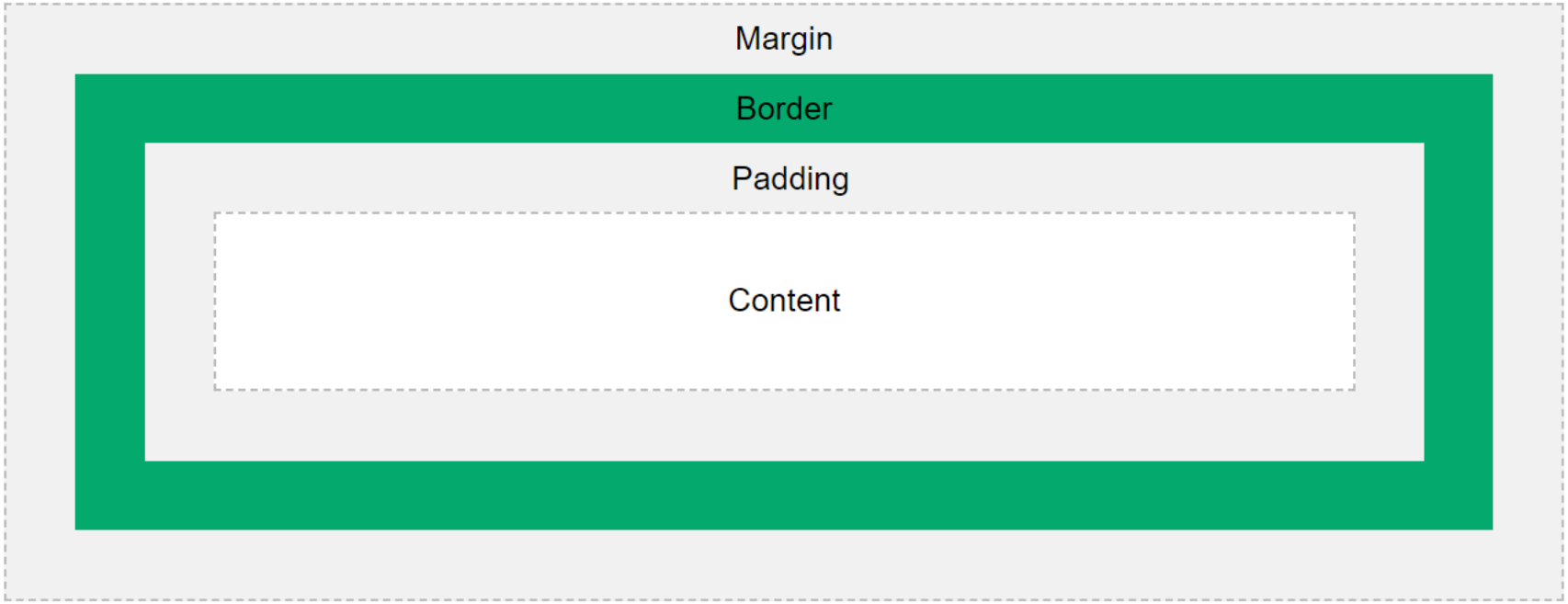
CSS Box Model

- In CSS, the term "box model" is used when talking about design and layout.
- The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content. The image below illustrates the box model:



CSS Box Model

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent



```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: skyblue;
  width: 200px;
  border: 15px solid red;
  padding: 30px;
  margin: 10px;
}
</style>
</head>
<body>

<h2>CSS Box Model</h2>

<p>The CSS box model example with borders, padding,
margins, and content.</p>

<div>division apply here</div>

</body>
</html>
```

CSS BOX Model Example

CSS Lists

In HTML, there are two main types of lists:

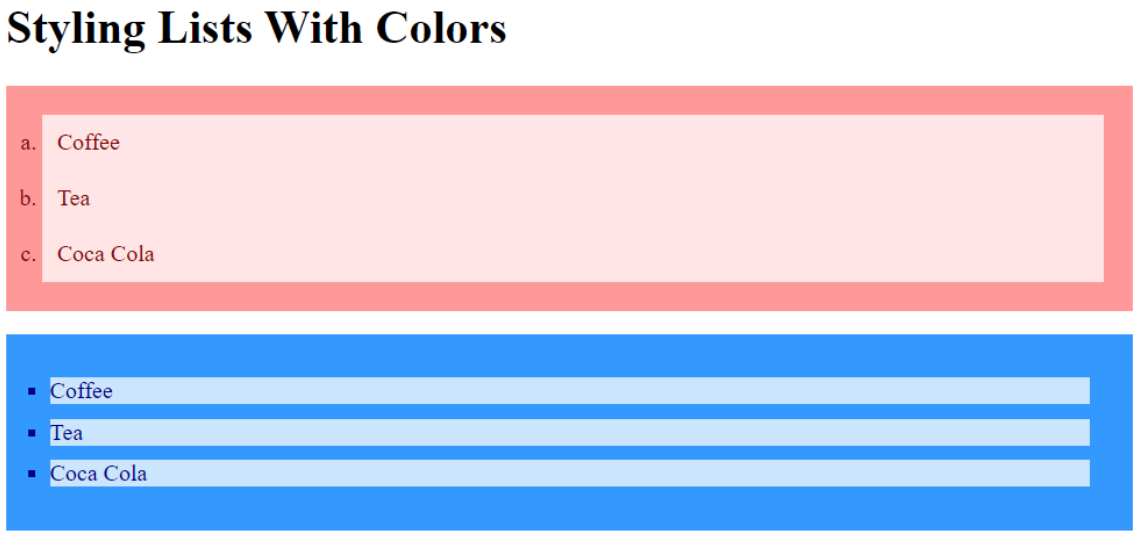
unordered lists (<ul>) - the list items are marked with bullets

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

```
<!DOCTYPE html>
<html>
<head>
<style>
ol {
  background: #ff9999;
  padding: 20px;
  list-style-type: lower-alpha
}

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

ol li {
  background: #ffe5e5;
  color: darkred;
  padding: 10px;
  margin-left: 5px;
}

ul li {
  background: #cce5ff;
  color: darkblue;
  margin: 10px;
  list-style-type: square
}
</style>
</head>
<body>

<h1>Styling Lists With Colors</h1>

<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

</body>
</html>
```



CSS Table

We can apply style on HTML tables for better look and feel. There are some CSS properties that are widely used in designing table using CSS:

- border
- border-collapse
- padding
- width
- height
- text-align
- color
- background-color

CSS Table

CSS Table Border

We can set border for the table, th and td tags using the CSS border property.

```
<style>
table, th, td {
    border: 1px solid black;
}
</style>
```

CSS Table Border Collapse

By the help of border-collapse property, we can collapse all borders in one border only.

```
<style>
table, th, td {
    border: 2px solid black;
    border-collapse: collapse;
}
</style>
```

CSS Table Padding

We can specify padding for table header and table data using the CSS padding property.

```
<style>
table, th, td {
    border: 1px solid black;
    border-collapse: collapse;
}
th, td {
    padding: 10px;
}
</style>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
#a {
  font-family: Arial, Helvetica,
     sans-serif;
  width: 100%;
}

#a td, #a th {
  border: 1px solid red;
  padding: 8px;
}

#a th {
  padding-top: 12px;
  padding-bottom: 12px;
  text-align: center;
  background-color: yellow;
  color: black;
}
</style>
</head>
<body>

<h1>CSS Table</h1>

<table id="a">
  <tr>
    <th>Roll No</th>
    <th>Name</th>
  </tr>
  <tr>
    <td>123</td>
    <td>Nitin</td>
  </tr>
  <tr>
    <td>546</td>
    <td>Piyush</td>
  </tr>


</table>

</body>
</html>
```

CSS Table

Roll No	Name
123	Nitin
546	Piyush

CSS Display

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

```
<!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="w1.html" target="_blank">HTML</a></li>
  <li><a href="w2.html" target="_blank">CSS</a></li>
  <li><a href="w3.html" target="_blank">JavaScript</a></li>
</ul>

</body>
</html>
```

Display a list of links as a horizontal menu:

HTML CSS JavaScript

CSS Display

The following example displays elements as block elements:

Example

```
span {
  display: block;
}
```

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:

Example

```
h1.hidden {
  display: none;
}
```

visibility:hidden; also hides an element.

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

Example

```
h1.hidden {
  visibility: hidden;
}
```

Pseudo Class Selectors

- A pseudo-class is used to define a special state of an element.
- For example, it can be used to:
  - Style an element when a user mouses over it
  - Style visited and unvisited links differently
  - Style an element when it gets focus

Pseudo Class Selectors

pseudo-class	Description
:active	It is used to add style to an active element.
:hover	It adds special effects to an element when the user moves the mouse pointer over the element.
:link	It adds style to the unvisited link.
:visited	It adds style to a visited link.
:focus	It selects the element which is focused by the user currently.
:first-child	It adds special effects to an element, which is the first child of another element.

CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element

All CSS Pseudo Classes Cont..

:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

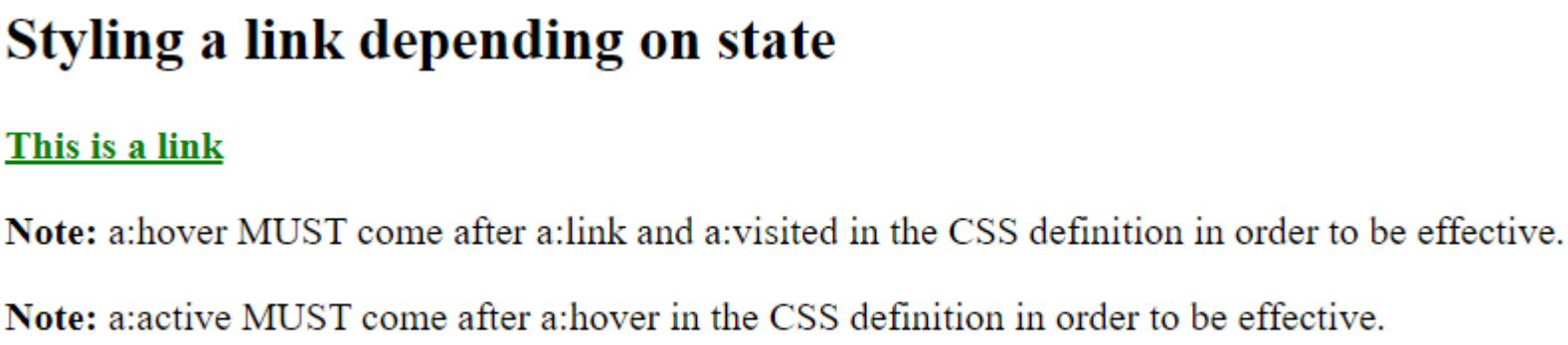
Anchor Pseudo class

```
/* unvisited link */
a:link {
  color: #FF0000;
}

/* visited link */
a:visited {
  color: #00FF00;
}

/* mouse over link */
a:hover {
  color: #FF00FF;
}

/* selected link */
a:active {
  color: #0000FF;
}
```

Here:

- a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!
- a:active MUST come after a:hover in the CSS definition in order to be effective!
- Pseudo-class names are not case-sensitive.

```
<!DOCTYPE html>
<html>
<head>
<style>
/* unvisited link */
a:link {
  color: red;
}

/* visited link */
a:visited {
  color: green;
}

/* mouse over link */
a:hover {
  color: hotpink;
}

/* selected link */
a:active {
  color: blue;
}
</style>
</head>

<body>

<h2>Styling a link depending on state</h2>

<p><b><a href="https://www.abes.ac.in/" target="_blank">This is a
link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS
definition in order to be effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order
to be effective.</p>

</body>
</html>
```



Pseudo Element Selector

Pseudo Element Selector

- A CSS pseudo-element is used to style specified parts of an element.
- For example, it can be used to:
  - Style the first letter, or line, of an element
  - Insert content before, or after, the content of an element
- Syntax

```
selector::pseudo-element {
  property: value;
}
```

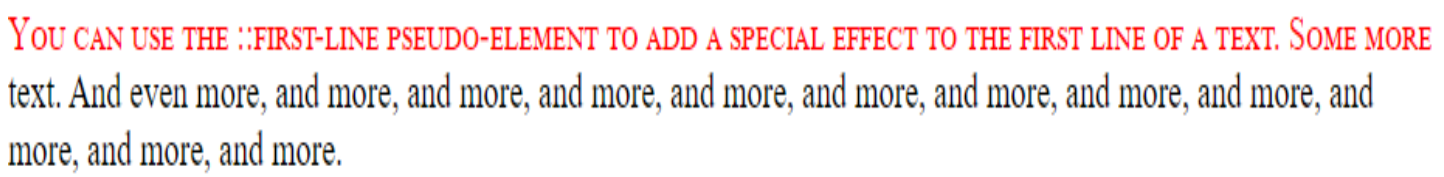
First Line Pseudo Element

- used to add a special style to the first line of a text.
- The following example formats the first line of the text in all <p> elements:
- Example

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

- The ::first-line pseudo-element can only be applied to block-level elements.

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
</style>
</head>
<body>

<p>You can use the ::first-line pseudo-element to add a special
effect to the first line of a text. Some more text. And even
more, and more, and more, and more, and more, and more,
and more, and more, and more, and more, and more, and
more.</p>

</body>
</html>
```



First Line Pseudo Element

The following properties apply to the ::first-line pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

Pseudo Element & HTML Classes

Pseudo-elements can be combined with HTML classes:

Example

```
p.intro::first-letter {
  color: #ff0000;
  font-size: 200%;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
p.intro::first-letter {
  color: #ff0000;
  font-size: 200%;
}
</style>
</head>
<body>

<p class="intro">This is an introduction.</p>
<p>This is a paragraph with some text. A bit more text
even.</p>

</body>
</html>
```

This is an introduction.

This is a paragraph with some text. A bit more text even.

The ::before Pseudo Element

- The ::before pseudo-element can be used to insert some content before the content of an element.
- The following example inserts an image before the content of each <h1> element:

```
h1::before {
  content: url(smiley.gif);
}
```

The ::after Pseudo Element

- used to insert some content after the content of an element.
- The following example inserts an image after the content of each <h1> element:

```
h1::after {
  content: url(smiley.gif);
}
```

The ::marker Pseudo Element

- selects the markers of list items.
- The following example styles the markers of list items:

```
::marker {
  color: red;
  font-size: 23px;
}
```

The ::selection Pseudo Element

- matches the portion of an element that is selected by a user.
- The following CSS properties can be applied to ::selection: color, background, cursor, and outline.
- The following example makes the selected text red on a yellow background:

```
::selection {
  color: red;
  background: yellow;
}
```

CSS: Border Radius

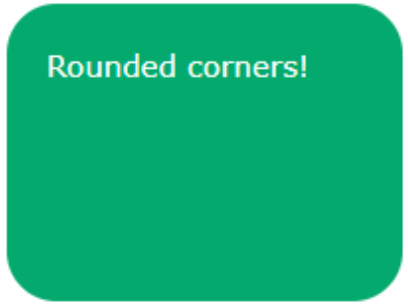
CSS Rounded Corners

- With the CSS border-radius property, you can give any element "rounded corners".
- defines the radius of an element's corners.
- The border-radius property is actually a shorthand property for the border-top-left-radius, border-top-right-radius, border-bottom-right-radius and border-bottom-left-radius properties.

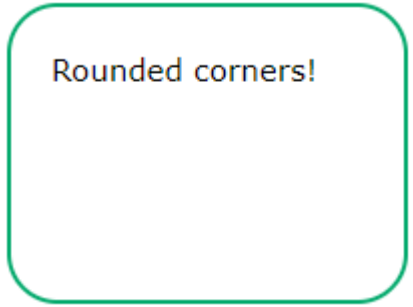
CSS Rounded Corners

- Here are three examples:

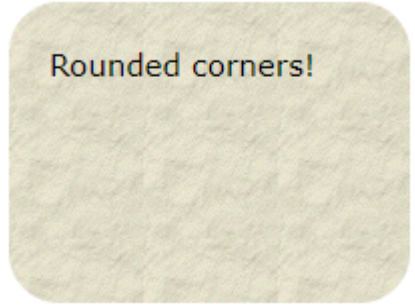
1. Rounded corners for an element with a specified background color:



2. Rounded corners for an element with a border:



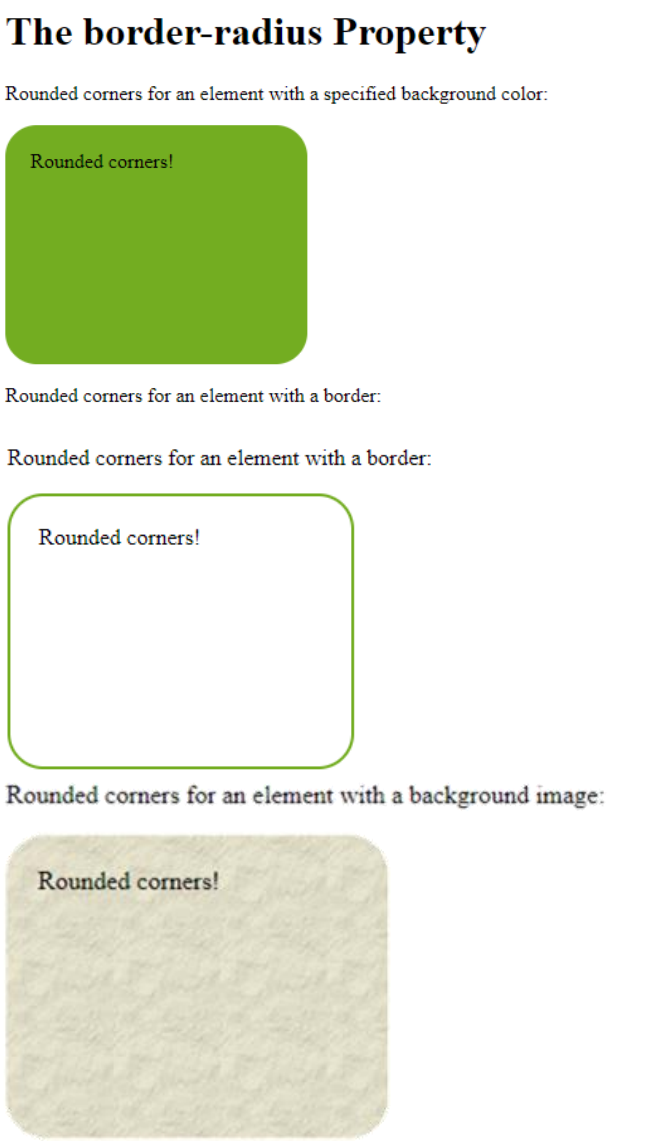
3. Rounded corners for an element with a background image:



CSS Rounded Corners

```
<!DOCTYPE html>
<html>
<head>
<style>
#rcorners1 {
  border-radius: 25px;
  background: #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}

#rcorners2 {
  border-radius: 25px;
  border: 2px solid #73AD21;
  padding: 20px;
  width: 200px;
  height: 150px;
}
#rcorners3 {
  border-radius: 25px;
  background: url(paper.gif);
  background-position: left top;
  background-repeat: repeat;
  padding: 20px;
  width: 200px;
  height: 150px;
}
</style>
</head>
<body>
```

```
<body>

<h1>The border-radius Property</h1>

<p>Rounded corners for an element with a specified background color:</p>
<p id="rcorners1">Rounded corners!</p>
<p>Rounded corners for an element with a border:</p>
<p id="rcorners2">Rounded corners!</p>
<p>Rounded corners for an element with a background image:</p>
<p id="rcorners3">Rounded corners!</p>

</body>
</html>
```



CSS border-radius - Specify Each Corner

The border-radius property can have from one to four values. Here are the rules:

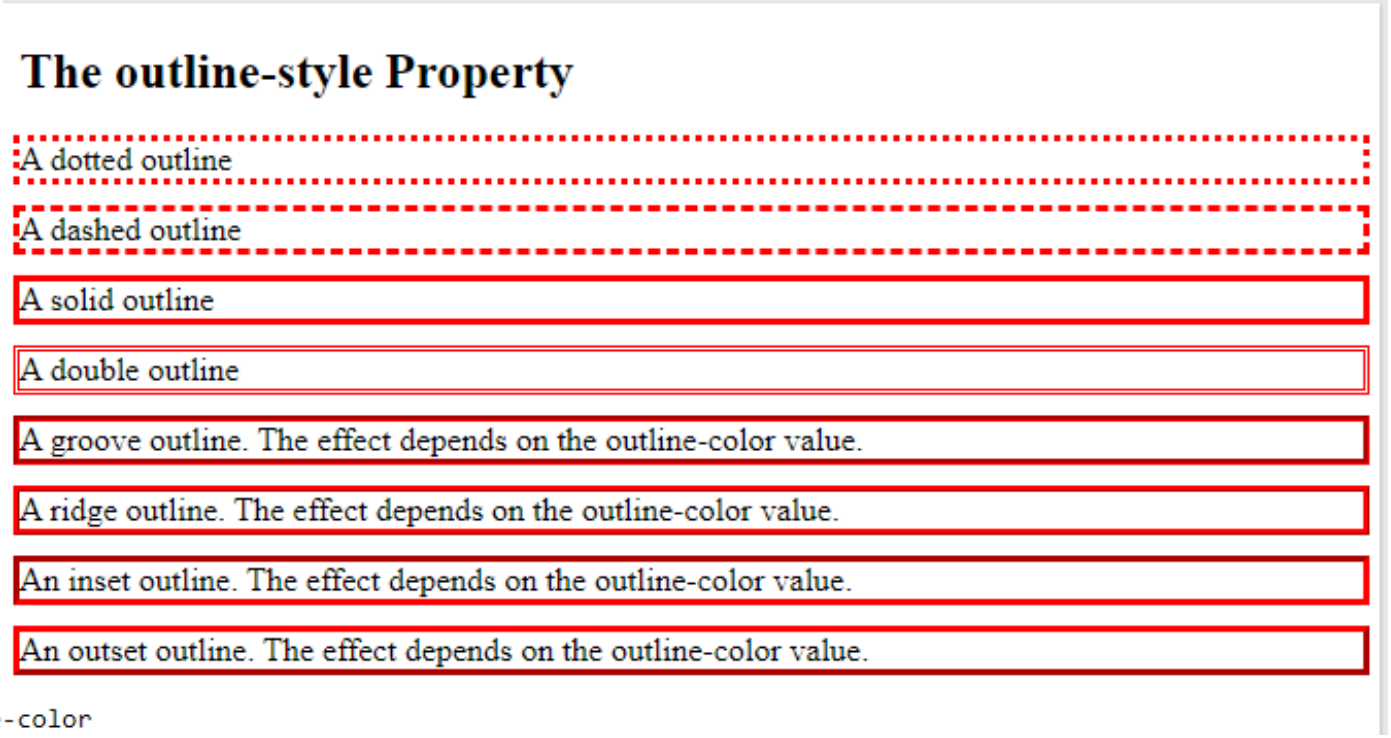
- Four values - border-radius: 15px 50px 30px 5px; (first value applies to top-left corner, second value applies to top-right corner, third value applies to bottom-right corner, and fourth value applies to bottom-left corner)
- Three values - border-radius: 15px 50px 30px; (first value applies to top-left corner, second value applies to top-right and bottom-left corners, and third value applies to bottom-right corner)

CSS border-radius - Specify Each Corner

- Two values - border-radius: 15px 50px; (first value applies to top-left and bottom-right corners, and the second value applies to top-right and bottom-left corners)
- One value - border-radius: 15px; (the value applies to all four corners, which are rounded equally)

CSS Outline Style

```
<!DOCTYPE html>
<html>
<head>
<style>
p {outline-color:red;}

p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
</style>
</head>
<body>

<h2>The outline-style Property</h2>
<p class="dotted">A dotted outline</p>
<p class="dashed">A dashed outline</p>
<p class="solid">A solid outline</p>
<p class="double">A double outline</p>
<p class="groove">A groove outline. The effect depends on the outline-color
value.</p>
<p class="ridge">A ridge outline. The effect depends on the outline-color
value.</p>
<p class="inset">An inset outline. The effect depends on the outline-color
value.</p>
<p class="outset">An outset outline. The effect depends on the outline-color
value.</p>

</body>
</html>
```



CSS Layout - The position Property

CSS Layout - The position Property

The position property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

The position Property

The position property specifies the type of positioning method used for an element.

There are five different position values:

- static
- relative
- fixed
- absolute
- sticky

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the position property is set first. They also work differently depending on the position value.

- **position: static;**

HTML elements are positioned static by default.
Static positioned elements are not affected by the top, bottom, left, and right properties.
An element with `position: static;` is not positioned in any special way; it is always positioned according to the normal flow of the page:

- **position: relative;**

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

position: relative;

An element with position: relative; is positioned relative to its normal position.

Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

- position: fixed;

An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.

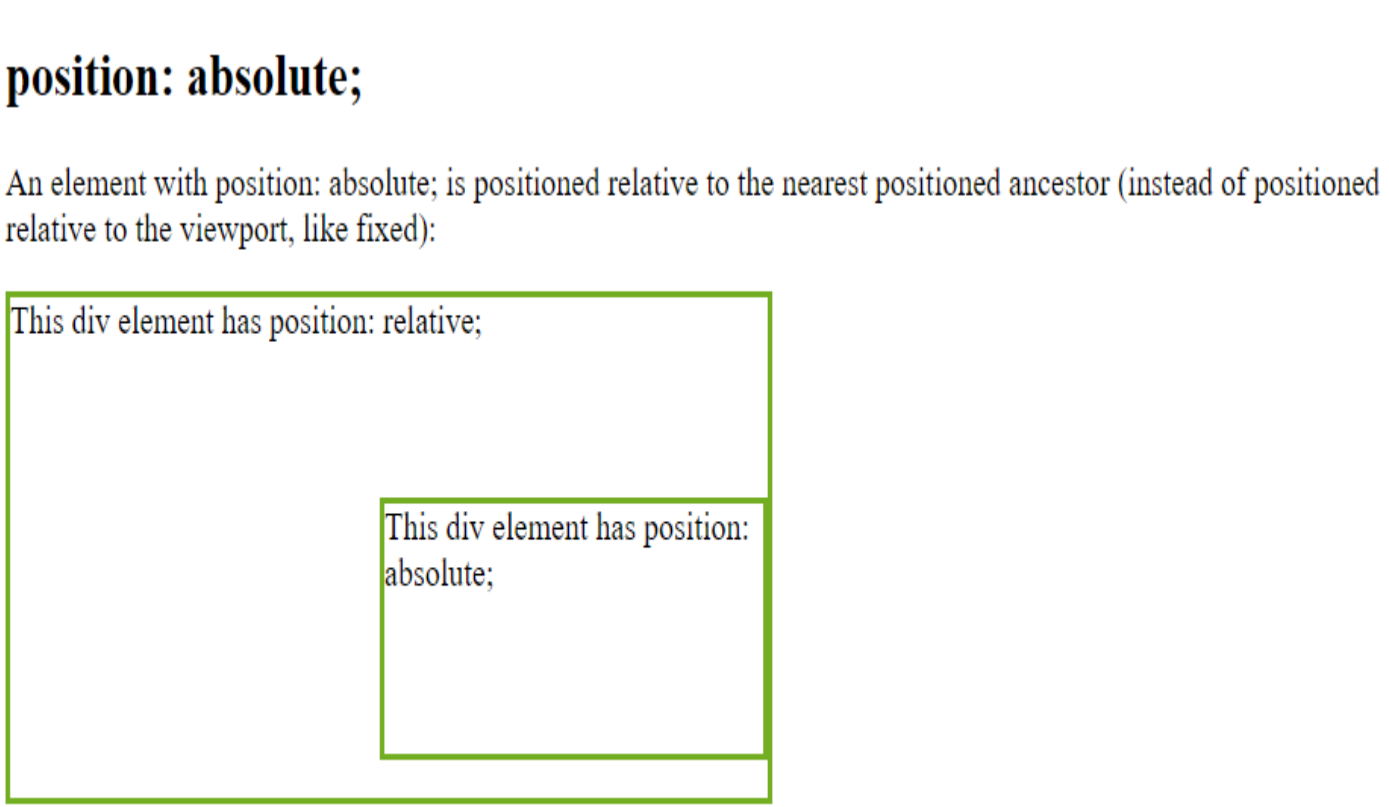
A fixed element does not leave a gap in the page where it would normally have been located.

- position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).

However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;
}

div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: absolute;</h2>

<p>An element with position: absolute; is positioned relative to the nearest positioned
ancestor (instead of positioned relative to the viewport, like fixed):</p>

<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>

</body>
</html>
```



CSS: Image Sprites

CSS Image Sprites

- An image sprite is a collection of images put into a single image.
- A web page with many images can take a long time to load and generates multiple server requests.
- Using image sprites will reduce the number of server requests and save bandwidth.

```
<!DOCTYPE html>
<html>
<head>
<style>
#navlist {
  position: relative;
}

#navlist li {
  margin: 0;
  padding: 0;
  list-style: none;
  position: absolute;
  top: 0;
}

#navlist li, #navlist a {
  height: 44px;
  display: block;
}

#home {
  left: 0px;
  width: 46px;
  background: url('img_navsprites.gif') 0 0;
}

#prev {
  left: 63px;
  width: 43px;
  background: url('img_navsprites.gif') -47px 0;
}

#next {
  left: 129px;
  width: 43px;
  background: url('img_navsprites.gif') -91px 0;
}
</style>
</head>
<body>

<ul id="navlist">
  <li id="home"><a href="https://www.abes.ac.in/"></a></li>
  <li id="prev"><a href="https://www.abes.ac.in/IT-department.php"></a></li>
  <li id="next"><a href="https://www.abes.ac.in/computer-science-
engineering.php"></a></li>
</ul>

</body>
</html>
```

Thank You