Module No. 1 – Part 2

Introduction to Web Design

Style sheets: Introduction CSS, Applying CSS to HTML, Selectors, Properties and Values, CSS Colors and Backgrounds, CSS Box Model, CSS Margins, Padding, and Borders, CSS Text and Font Properties.

Cascading Style Sheets (CSS)

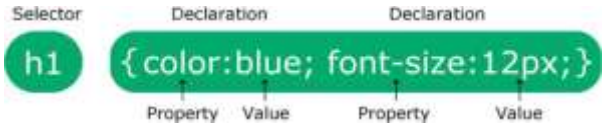
- CSS stands for Cascading Style Sheets.
- HTML is a markup language used to create static web pages and web applications, whereas CSS is a style sheet language responsible for the presentation of documents written in a markup language.
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media. (CSS is the language we use to style an HTML document).

Difference between HTML and CSS

HTML	CSS
HTML is a markup language used to define a ***structure of a web page***.	CSS is a style sheet language used to ***style the web pages*** by using different styling features.
It consists of ***tags*** inside which text is enclosed.	It consists of ***selectors and declaration blocks.***
HTML does not have further types.	CSS can be ***internal or external*** depending upon the requirement.
We ***cannot use HTML inside a CSS sheet***.	We can ***use CSS inside an HTML*** document.
HTML is not used for presentation and visualization.	CSS is used for ***presentation and visualization.***
HTML does not allow animations and transitions.	CSS ***allows animation and transitions*** which helps to improve the UI.
HTML files are saved with ***.htm*** or ***.html*** extension.	CSS files are saved with ***.css*** extension.

CSS Syntax

- ***A CSS rule consists of a selector and a declaration block***.



- The ***selector points to the HTML element you want to style***.
- The declaration block contains one or more declarations separated by semicolons.
- Each declaration includes a CSS property ***name and a value, separated by a colon***.
- Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  color: red;
  text-align: center;
}
</style>
</head>

<body>
<p>This Program is styled using CSS!!!</p>
</body>
</html>
```

This Program is styled using CSS!!!

Example Explained

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value.
- text-align is a property, and center is the property value.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
        background-color: rgba(0, 0, 255, 0.5);
   }
h1 {
        color: orange;
        text-align: center;
   }
p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>

<body>
<h1> Hello Everyone :-) </h1>
<p>This Program is styled using CSS!!!</p>
</body>
</html>
```

CSS Selectors

- CSS selectors are ***used to "find" (or select) the HTML elements you want to style***.
- We can divide CSS selectors into five categories:
    - **Simple selectors:** select elements based on name, id, class.
    - **Combinator selectors:**
    - **Pseudo-class selectors:**
    - **Pseudo-elements selectors:**
    - **Attribute selectors**:

The CSS element Selector:

- The element selector selects HTML elements ***based on the element name***.

Example-1:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
   fony-family: Calibri (Body);
   font-size: 30px;
   text-align: center;
   color: blue;
  }
</style>
</head>
```

```
<body>
<p> VIT-AP :-) </p>
<p> Amaravati!! </p>
<h1> Andhra Pradesh </h1>
</body>
</html>
```

Example-2:

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

h1 {
   text-align: center;
   color: green;
}
</style>
</head>
```

```
<body>
<p> VIT-AP :-) </p>
<p> Amaravati!! </p>
<h1> Andhra Pradesh </h1>
</body>
</html>
```

The CSS id Selector:

- The id selector ***uses the id attribute of an HTML element to select a specific element***.
- The ***id of an element is unique within a page***, so the id selector is used to select one unique element and Not reusable.
- To select an element with a specific id, ***write a hash (#) character, followed by the id of the element***.

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

</style>
</head>

<body>
<p> VIT-AP :-) </p>
<p> Amaravati!! </p>
</body>
</html>
```

VIT-AP :-)

Amaravati!!

Example: The CSS rule below will be applied to the HTML element with id="para1"

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

</style>
</head>
```

```
<body>
<p id = "para1"> VIT-AP :-) </p>
<p id = "para1"> Amaravati!! </p>
<h1 id = “para1”> Andhra Pradesh </h1>
</body>
```

VIT-AP :-)

Amaravati!!

The CSS class Selector:

- The class ***selector selects HTML elements with a specific class attribute***.
- To select elements with a specific class, ***write a period (.) character, followed by the class name***. Can be used on **multiple elements** and **Highly reusable.**

Example-1: In this example, all HTML elements with class="center" will be blue and center-aligned, is used in both **h1** and **p**.

```
<!DOCTYPE html>
<html>
<head>
<style>
.center {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

</style>
</head>

<body>
<h1 class="center"> VIT-AP-Amaravati :-) </h1>
<p class="center"> Andhra Pradesh!!! </p>
</body>
</html>
```

Example-2: You can also specify that only specific HTML elements should be affected by a class.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

</style>
</head>
```

```
<body>
<h1 class="center"> VIT-AP-Amaravati :-) </h1>
<p class="center"> Andhra Pradesh!!! </p>
</body>
</html>
```

Example-3 : In this example only <h1> elements with class="center".

```
<!DOCTYPE html>
<html>
<head>
<style>
h1.center {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

</style>
</head>
```

```
<body>
<h1 class="center"> VIT-AP-Amaravati :-) </h1>
<p class="center"> Andhra Pradesh!!! </p>
</body>
</html>
```

Example-4: HTML elements can also refer to more than one class.

- In this example the <p> element will be styled according to class="center" and to class="large":

```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}
p.large{
            font-size: 500%;
        }

</style>
</head>
```

```
<body>
<h1 class="center"> VIT-AP-Amaravati :-) </h1>
<p class="center"> Andhra Pradesh!!! </p>
<p class="center large"> INDIA </p>
</body>
</html>
```

<u>The CSS Universal Selector:</u>

The ***universal selector (*) selects all HTML elements on the page***.

Example:

- The CSS rule below will affect every HTML element on the page:

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

* {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

</style>
</head>
```

```
<body>
<h1> Hello </h1>
<p> This is VIT-AP-Amaravati :-) </p>
<p> Andhra Pradesh!!! </p>
<p> INDIA </p>
</body>
</html>
```

Hello

This is VIT-AP-Amaravati :-)

Andhra Pradesh!!!

INDIA

The CSS Grouping Selector:

- The grouping selector selects all the HTML elements with the same style definitions.
- Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

Example:

```
h1 {
        text-align: center;
        color: blue;
   }
```

```
h2 {
        text-align: center;
        color: green;
   }
```

```
p {
        text-align: center;
        color: orange;
   }
```

- It will be better to group the selectors, to minimize the code.
- To group selectors, separate each selector with a comma.

Example-1: In this example we have grouped the selectors from the code above.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1,h2,p {
   font-family: Calibri;
   font-size: 30px;
   text-align: center;
   color: blue;
}

</style>
</head>
```

```
<body>
<h1> Hello :-) </h1>
<h2> This is VIT-AP-Amaravati </h2>
<p> Andhra Pradesh!!! </p>
<p> INDIA </p>
</body>
</html>
```

Hello

This is VIT-AP-Amaravati :-)

Andhra Pradesh!!!

INDIA

Example-2:

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

h1 {
   text-align: left;
   color: red;
}
h2 {
   text-align: center;
   color: green;
}
p {
   text-align: right;
   color: blue;
}

</style>
</head>
```

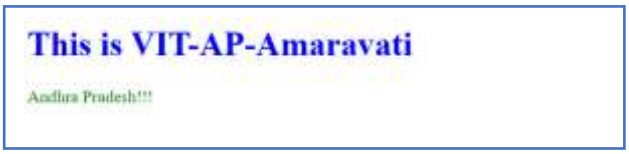
```
<body>
<h1> Hello :-) </h1>
<h2> This is VIT-AP-Amaravati </h2>
<p> Andhra Pradesh!!! </p>
<p> INDIA </p>
</body>
</html>
```

How To Add CSS (or) Methods to Link CSS to HTML

- Adding CSS (**Cascading Style Sheets**) to your HTML is ***essential for creating visually appealing and user-friendly web pages.***
- CSS can be added to HTML documents in 3 ways (or) There are three ways of inserting a style sheet:
    - Inline CSS
    - Internal CSS
    - External CSS

- **Inline Styles:**
    - An inline CSS is used to apply a ***unique style to a single HTML element.***
    - By using the style attribute inside HTML elements.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="color:blue"> This is VIT-AP-Amaravati </h1>
<p style="color:green"> Andhra Pradesh!!! </p>

</body>
</html>
```



- **Internal CSS:**
  - An internal CSS is ***used to define a style for a single HTML page***.
  - By using a <style> element in the <head> section.

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

body { background-color: lightgray; }
h1 { color: blue; }
p { color: green; }

</style>
</head>
```

```
<body>
<h1> This is VIT-AP-Amaravati </h1>
<p> Andhra Pradesh!!! </p>

</body>
</html>
```

- **External CSS:**
  - An external style sheet is ***used to define the style for many HTML pages***.
  - By using a <link> element to link to an external CSS file.
  - **rel** Specifies the relationship between the current document and the linked document and **href** Specifies the location of the linked document.

External_css.css

```
body {
   background-color: lightgray;
}

h1 {
   color: blue;
}

p {
   color: green;
}
```

```
<!DOCTYPE html>
<html>
<head>
   <link rel="stylesheet" href="external_css.css">
</head>

<body>
   <h1> This is VIT-AP-Amaravati </h1>
   <p> Andhra Pradesh!!! </p>
</body>
</html>
```

Embedded Style Sheets

- An embedded style sheet will allow you to ***address multiple HTML elements at once***, but an inline style will only allow you to address one HTML element at a time.
- Embedded style sheets refer to ***when you embed style sheet information into an HTML document using the <style> element***. You do this by embedding the style sheet information ***within <style></style> tags*** in the head of your document.

```
<!DOCTYPE html>
<html>
<head>
   <style>
      body {
         background-color: lightyellow;
      }
      h1 {
         color: blue;
         text-align: center;
      }
      p {
         color: green;
         font-size: 18px;
      }
   </style>
</head>
```

```
<body>
   <h1> VIT-AP University </h1>
   <p> Located in Amaravati, Andhra Pradesh. </p>
</body>
</html>
```

Conflicting Styles

- **There are several ways that conflicting styles can arise in CSS. One common cause is using multiple stylesheets, either external or internal, that have overlapping or conflicting rules. When styles from multiple stylesheets are applied to the same element, the browser will have to decide which style to use, leading to conflicts.**
- Another cause of conflicting styles is the use of multiple selectors that apply to the same element. If these selectors have conflicting styles, the browser will have to decide which style to apply, based on the specificity and order of the selectors.

Example: In this example, we have two conflicting styles for the color property applied to the same element **p** and **.special**. However, the **.special** selector has a higher specificity than the **p** selector, so the color: green style will override the color: blue style.

```
<!DOCTYPE html>
<html>
<head>
<style>
p {
        color: green;
  }
.special {
        color: blue;
         }
</style>
</head>
```

```
<body>

<p class="special">This is VIT-AP-Amaravati-Andhra Pradesh!!! </p>

</body>
</html>
```

This is VIT-AP-Amaravati-Andhra Pradesh!!!

Example 2: Inheritance is the process by which styles applied to a parent element are passed down to its child elements. If two selectors have the same specificity, the one that comes later in the stylesheet will override the earlier one. If one of the conflicting styles is inherited, the browser will use the inherited style over the non- inherited style, even if the non-inherited style has a higher specificity or is defined later in the cascade order.

```
<!DOCTYPE html>
<html>
<head>
<style>
/*style applied to the parent element*/
.parent  {
        font-size: 20px;
        color:blue;
  }
/*conflicting style applied to the child element*/
.child  {

  }
</style>
```

```
<body>

<div class ="parent">
<p> Parent Class </p>
<div class ="child">
<p> Child Class </p>
</div>
</div>

</body>
</html>
```

Parent Class

Child Class

Example 3:

- If the child **does not define a property**, it inherits from the parent
- If the child **defines the same property**, it **overrides** the parent
- The <p> inside .parent **inherits**: Because <p> has **no own CSS**, it takes from its parent.
- This child is **nested inside** `.parent`

- So initially, it **inherits**:
  - `font-size: 20px`
  - `color: blue`

👉 This is **inheritance**.

Now:

- font-size → **overridden to 14px**
- color → **overridden to red**

👉 Because the child **explicitly defines these properties**, inheritance stops for them.

```
<!DOCTYPE html>
<html>
<head>
<style>
/*style applied to the parent element*/
.parent {
   font-size: 20px;
   color: blue;
}
/*conflicting style applied to the child element*/
.child {
   font-size: 14px;
   color: red;

}
</style>
```

```
<body>

<div class="parent">
   <p>Parent Class</p>

   <div class="child">
      <p>Child Class</p>
   </div>
</div>

</body>
</html>
```

Positioning Elements

- CSS positioning defines ***how elements are placed within a web page***. It allows you to ***control the layout, stacking order, and alignment of elements***.
- The primary positioning types in CSS are:

Position property	Description
Fixed	An element with position: fixed property remains in the same position relative to the viewport even when the page is scrolled.
Static	Default positioning method. Elements with position: static are positioned according to the normal flow of the document.
Relative	Elements with position: relative are positioned relative to their normal position in the document flow. Other elements will not fill the gap left by this element when adjusted.
Absolute	Positioned concerning its nearest non-static ancestor. Elements with position: absolute are taken out of the normal document flow.
Sticky	Combines features of position: relative and position: fixed. The element is treated as position: relative until it reaches a specified threshold, then it becomes position: fixed.

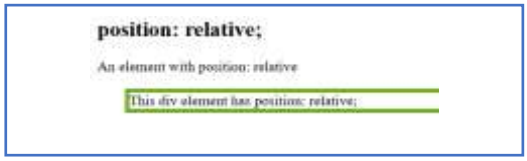
Relative Positioning:

- 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.

Example-1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
   position: relative;
   left: 30px;
   border: 5px solid #73AD21;
}
</style>
</head>
```

```
<body>
<h2> position: relative; </h2>
<p> An element with position: relative </p>

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

</body>
</html>
```



Example-2:

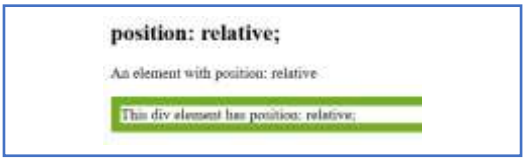
```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
   position: relative;
   left: 10px;
   border: 5px solid #73AD21;
}
</style>
</head>
```

```
<body>
<h2> position: relative; </h2>
<p> An element with position: relative </p>

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

</body>
</html>
```

Example-3:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
   position: relative;
   left: 1px;
   border: 10px solid #73AD21;
}
</style>
</head>
```

```
<body>
<h2> position: relative; </h2>
<p> An element with position: relative </p>

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

</body>
</html>
```



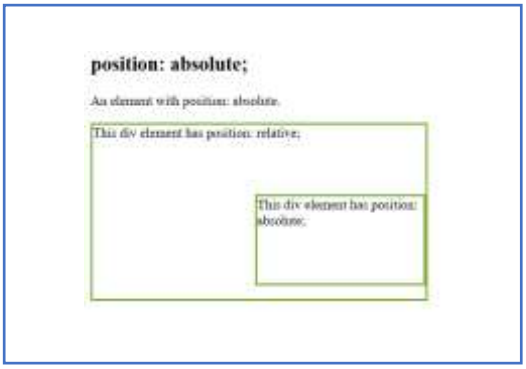
Absolute Positioning:

- 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.
- **Note:** Absolute positioned elements are removed from the normal flow, and can overlap elements.

Example 1:

```
<!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. </p>

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

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

</body>
</html>
```



① Parent DIV – position: relative

Think of this **parent div** as a **box on the page**.

- It stays in its normal place
- It **creates a reference point**
- It does **not move** anywhere

👉 Important rule: A relative element acts as a **reference (boundary)** for absolute elements inside it.

② **Child DIV – position: absolute**

The **child div**:

- Is removed from normal flow
- Is positioned **inside the parent box**
- Uses top and right values

👉 Meaning:

- top: 80px → move 80px down from top of parent
- right: 0 → stick to the right edge of parent

Why absolute element stays inside the parent? →

```
.parent {
   position: relative;
}
```

✓ Absolute element always looks for the **nearest positioned parent**

✓ Here, .relative is that parent

✓ So the child stays **inside the green border**

🏠 **House Example**

- **Relative div** → House
- **Absolute div** → Furniture inside the house

📌 Furniture is placed **relative to the house walls**, not the whole city.

z-index

- The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).
- An element with greater stack order is always in front of an element with a lower stack order. z-index decides the **depth** of elements (front or back)
- **Note:** z-index only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display:flex elements).

Without z-index: If two positioned elements overlap each other without a z-index specified, the element defined **last in the HTML code** will be shown on top.

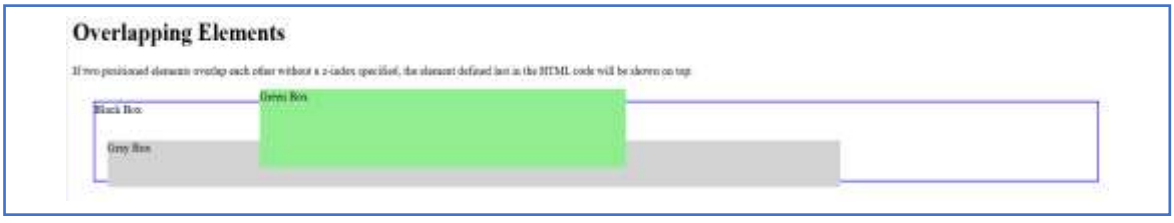
```
<!DOCTYPE html>
<html>
<head>
<style>
.container {
   position: relative;
}
```

```
/* Black box */
.black-box {
   position: relative;
   border: 2px solid blue;
   height: 100px;
   margin: 30px;
}
```

```
/* Gray box */
.gray-box {
   position: absolute;
   background: lightgray;
   height: 60px;
   width: 70%;
   left: 50px;
   top: 50px;
}
```

```
/* Green box */
.green-box {
   position: absolute;
   background: lightgreen;
   width: 35%;
   left: 270px;
   top: -15px;
   height: 100px;
}
</style>
</head>
```

```
<body>
<h1> Overlapping Elements </h1>
<p>
If two positioned elements overlap each other without a z-index specified,
the element defined last in the HTML code will be shown on top:
</p>

<div class="container">
   <div class="black-box"> Black Box </div>
   <div class="gray-box"> Gray Box </div>
   <div class="green-box"> Green Box </div>
</div>

</body>
</html>
```



<u>With z-index:</u> Higher z-index → comes in front & Lower z-index → goes behind

A **container** is used to **hold and control other elements**. Keeps related items together and Easy to manage as one unit

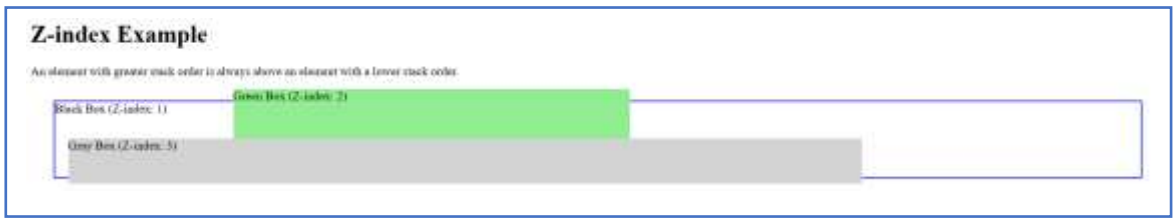
```
<!DOCTYPE html>
<html>
<head>
<style>
.container {
   position: relative;
}
```

```
/* Black box */
.black-box {
   position: relative;
   z-index: 1;
   border: 2px solid blue;
   height: 100px;
   margin: 30px;
}
```

```
/* Gray box */
.gray-box {
   position: absolute;
   z-index: 3;
   background: lightgray;
   height: 60px;
   width: 70%;
   left: 50px;
   top: 50px;
}
```

```
/* Green box */
.green-box {
   position: absolute;
   z-index: 2;
   background: lightgreen;
   width: 35%;
   left: 270px;
   top: -15px;
   height: 100px;
}
</style>
</head>
```

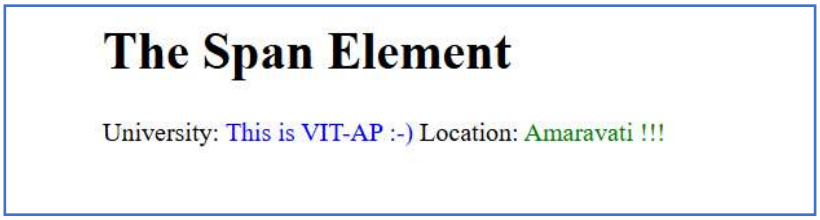
```
<body>
<h1> Z-index Example </h1>
<p>
An element with greater stack order is always above an element with a lower stack order.
</p>

<div class="container">
   <div class="black-box"> Black Box (Z-index: 1) </div>
   <div class="gray-box"> Gray Box (Z-index: 3) </div>
   <div class="green-box"> Green Box (Z-index: 2) </div>
</div>

</body>
</html>
```



Span

- is used when we want to **highlight or style a word or a group of words** inside a sentence.
- is used to apply style or effect to a small part of text without breaking the line.
- The tag is easily styled by CSS or manipulated with JavaScript using the class or id attribute.
- The tag is much like the element, but *** is a block-level element and is an inline element***.

```
<!DOCTYPE html>
<html>
<body>

<h1> The Span Element </h1>
<p>
University: <span style = color:blue;> This is VIT-AP :-) </span>
Location: <span style = color:green;> Amaravati !!! </span>
</p>

</body>
</html>
```



Example:

```
<p>
  Welcome to <span class="college">VIT-AP</span> University
</p>
.college {
   color: blue;
   font-weight: bold;
}
```

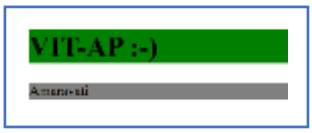
CSS Colors

- Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.
- **CSS Color Names:** In CSS, a color can be specified by using a predefined color name.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="background-color:Tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
<h1 style="background-color:Grey;">Grey</h1>
<h1 style="background-color:SlateBlue;">SlateBlue</h1>
<h1 style="background-color:Violet;">Violet</h1>
<h1 style="background-color:LightGrey;">LightGrey</h1>

</body>
</html>
```

CSS Background Color: You can set the background color for HTML elements.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="background-color:Green;">VIT-AP :-)</h1>
<p style="background-color:Grey;">Amaravati</p>

</body>
</html>
```

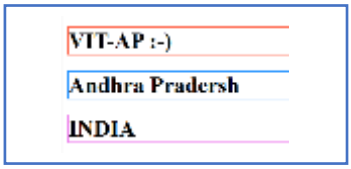


CSS Text Color: You can set the color of text.

```
<!DOCTYPE html>
<html>
<body>

<h3 style="color:Green;">VIT-AP :-)</h3>
<p style="color:Blue;">Amaravati</p>
<p style="color:Orange;">INDIA</p>

</body>
</html>
```

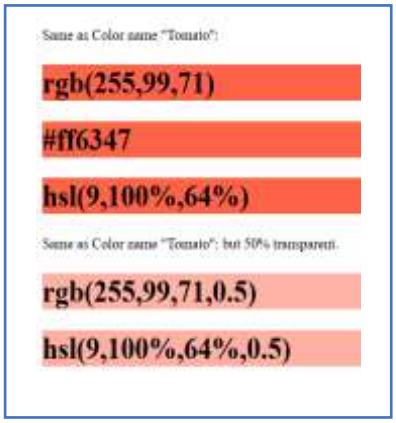
CSS Border Color: You can set the color of borders.

```
<!DOCTYPE html>
<html>
<body>

<h1 style="border:2px solid Tomato;">VIT-AP :-)</h1>
<h1 style="border:2px solid DodgerBlue;">Andhra Pradersh</h1>
<h1 style="border:2px solid Violet;">INDIA</h1>

</body>
</html>
```



CSS Color Values: In CSS, colors can also be specified using RGB values, HEX values, HSL values, RGBA values, and HSLA values.

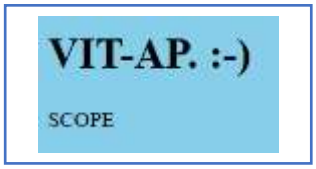
```
<!DOCTYPE html>
<html>
<body>

<p> Same as Color name "Tomato":</p>
<h1 style="background-color:rgb(255,99,71);">rgb(255,99,71)</h1>
<h1 style="background-color:#ff6347;">#ff6347</h1>
<h1 style="background-color:hsl(9,100%,64%);">hsl(9,100%,64%)</h1>

<p> Same as Color name "Tomato": but 50% transparent.</p>
<h1 style="background-color:rgb(255,99,71,0.5);">rgb(255,99,71,0.5)</h1>
<h1 style="background-color:hsla(9,100%,64%,0.5);">hsl(9,100%,64%,0.5)</h1>

</body>
</html>
```



CSS background-color: The background-color property specifies the background color of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
   background-color: skyblue;
}
</style>
</head>
```

```
<body>
<h1>VIT-AP. :-)</h1>
<p>SCOPE</p>
</body>

</html>
```

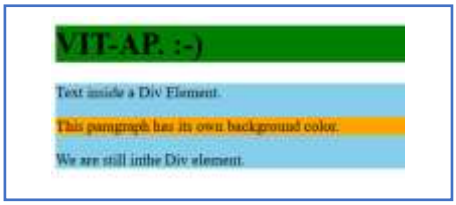


Other Elements: You can set the background color for any HTML elements:

- Here, the <h1>, <p>, and elements will have different background colors:
- The tag defines a division or a section in an HTML document

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
   background-color: green;
}
div {
   background-color: skyblue;
}
p {
   background-color: orange;
}
</style>
</head>
```

```
<body>
<h1>VIT-AP. :-)</h1>

<div>Text inside a Div Element.
<p> This paragraph has its own background color. </p>
We are still in the Div element.
</div>

</body>

</html>
```



CSS background-image:

- The background image property specifies an image to use as the background of an element.
- By default, the image is repeated so it covers the entire element.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
   background-image: url("VIT-AP_University_seal.png");   /* image file */
    background-repeat: no-repeat;
}
</style>
</head>
```

```
<body>
<h1>VIT-AP. :-)</h1>
<p>SCOPE</p>
</body>

</html>
```

CSS background-repeat:

- By default, the background-image property repeats an image both horizontally and vertically.
- Some images should be repeated only horizontally or vertically, or they will look strange, like this

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
        background-image: url("VIT-AP_University_seal.png");
        background-repeat: repeat;
}
</style>
</head>
```

```
<body>
<h1>VIT-AP. :-)</h1>
<p>SCOPE</p>
<p> Repeat the Image both
Horizantally and Vertically </p>
</body>

</html>
```

- If the image above is repeated only horizontally (**background-repeat: repeat-x;**)
- To repeat an image vertically, set **background-repeat: repeat-y**;
- Showing the background image only once is also specified by the **background-repeat: no-repeat;**

Element Dimensions

CSS dimension defines the size and space occupied by elements on a webpage. The dimension properties like height, width, max-height, max-width, line-height and many more are used to define width, height of HTML elements in every screen sizes.

Dimension Properties:

- **Height:** Sets the fixed height of an element. The element will not grow or shrink beyond this height.
- **Width:** Sets the fixed width of an element.
- **max-height:** Sets the maximum height an element can grow to. The element can be smaller, but not taller than this value.
- **max-width:** Sets the **maximum width** an element can grow to.
- **min-height:** Sets the **minimum height** an element must have. Even with little or no content, height won't go below this value.
- **min-width:** Sets the **minimum width** an element must have.

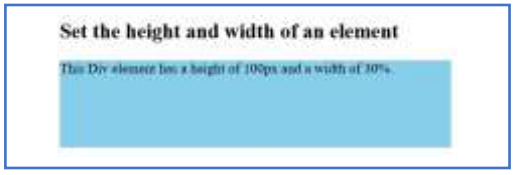
Height and Width Values: The CSS height and width properties might have the following values:

Value	Explanation
Auto	Default value. Browser automatically decides height and width
Length	Size given in fixed units like px, cm, etc.
%	Size is a percentage of the **parent (container)** element
Initial	Resets height/width to browser's default value
inherit	Takes the height/width from the parent element

Example-1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
        height:100px;
        width: 30%;
        background-color: skyblue;
   }
</style>
</head>
```

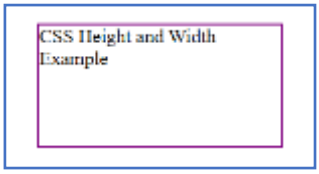
```
<body>
<h2>Set the height and width of an element </h2>
<div> This Div element has a height of 100px and a
width of 30%. </div>
</body>
</html>
```



Example-2:

```
<!DOCTYPE html>
<html>
<head>
<style>
.box {
   width: 200px;
   height: 100px;
   max-width: 300px;
   min-height: 80px;
   border: 2px solid purple;
}
</style>
</head>
```

```
<body>
<div class="box">
CSS Height and Width Example
</div>
</body>
</html>
```



- max-width is NOT used to make elements larger. It is used to STOP elements from becoming too wide on large screens.
- On **mobile** → small screen
- On **desktop / large monitor** → very wide screen
- 👉 The element stretches **too wide** on large screens

  👉 Text becomes hard to read
- On **small screens** → element shrinks normally
- On **large screens** → element stops growing at 300px
- max-width is used to restrict the maximum width of an element to maintain layout control and responsiveness across different screen sizes.

Box Model

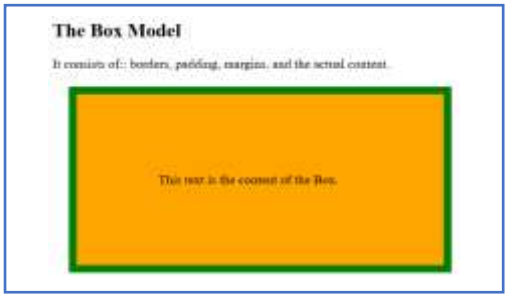
- In CSS, the term "box model" is used when talking ***about design and layout***.
- The CSS box model is essentially a box that wraps around every HTML element.
- **It consists of: content, padding, borders, and margins.**
- Explanation of the different parts:
    - **Content -** The content of the box, where text and images appear.
    - **Padding -** Padding is the space inside the element, between the content and the border. It creates **inner spacing**.
    - **Border -** A border that goes around the padding and content.
    - **Margin -** Margin is the space outside the element, between the element and other elements. It creates **outer spacing**. Margin keeps distance **between boxes**.
- The box model allows us to add a border around elements, and to define space between elements

Example-1:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: orange;
  width: 250px;
  border: 10px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
```

```
<body>
<h2> The Box Model </h2>
<p> It consists of:: borders, padding, margins, and the actual content. </p>
<div> This text is the content of the Box. </div>
</body>
</html>
```

Example-2: If the padding is 100px then observe the following output.



Text Flow

- Floating allows you to ***move an element to one side of the screen***; other content in the document then flows around the floated element.
- The CSS float property is a positioning property. It is ***used to push an element to the left or right, allowing other element to wrap around it***. It is generally used with images and layouts.

```
<!DOCTYPE html>
<html>
<head>
<title> Flowing text around Floating
Elements </title>
<style type = "text/css">
header {
   background-color: skyblue;
   text-align: center;
   font-family: arial, helvetica, sans-serif;
   padding: .2em;
  }
p {
   text-align: justify;
   font-family: verdana, geneva, sans-serif;
   margin: .5em;
  }
h1{
   margin-top: 0px;
   }
.floated {
   background-color: lightgrey;
   font-size: 1.5em;
   font-family: arial, helvetica, sans-serif;
   padding: .2em;
   margin-left: .5em;
   margin-bottom: .5em;
   float: right;
   text-align: right;
   width: 50%;
  }
Section {
   border:1px solid skyblue;
    }
</style>
</head>
```

```
<body>
<header><img src="VIT-AP_University_seal.png" alt = "VIT-AP"> </header>
<section>
<h1 class="floated"> SCOPE </h1>
<p> CSE, CSE(AI &ML), CSE(Blockchain), CSE(Cyber Security) </p>
<p> CSE(Data Analytics), CSE(Software Engineering), CSBS </p>
</section>
</body>
</html>
```

1. At the top, there is a header with an image (logo).
2. Below the header, there is a box (section).
3. Inside the box:
    - A heading SCOPE is placed on the right side.
    - Text appears on the left and below, flowing around the heading.

👉 *This is called a floating layout.*

- **Header:** The header is just a top bar with a background color and an image in the center.
    - background-color → gives color
    - text-align: center → centers the image
    - padding → gives space inside
- **Paragraphs:** Paragraphs contain normal text and are aligned neatly.
    - text-align: justify → text looks straight on both sides
    - margin → gives space between paragraphs
- **Floating Heading:** The heading **SCOPE** is not in the normal position. It is **pushed to the right** using float: right.
    - float: right → moves the **heading box** to the right
    - width: 50% → heading takes **half the page**
    - Text **wraps around** the heading automatically

📌 **Key sentence:** Float moves the box, not just the text.

- **Difference Between float and text-align:** simple comparison
    - float: right → moves the **whole box**
    - text-align: right → moves the **text inside the box**

Media Types

- One of the most important features of style sheets is that they specify **how a document is to be presented on different media**: on the screen, on paper, with a speech synthesizer, with a braille device, etc.
- We have currently **two ways to specify media dependencies for style sheets** –
    - Specify the target medium from a style sheet with the **@media** or **@import at-rules**.
    - Specify the target medium within the document language.

- The @media rule: this rule specifies the target media types (separated by commas) of a set of rules.

```
<style type = "text/css">
  <!--
     @media print {
                    body { font-size: 10pt }
          @media screen {
                    body { font-size: 12pt }
          @media screen, print {
                    body { line-height: 1.2pt }
   -->
</style>
```

- **The Document Language:** In HTML, the media attribute on the LINK element specifies the target media of an external style sheet –

```
<!DOCTYPE html>
<html>
  <head>
    <title>Link to target medium</title>
          <link rel="stylesheet" type="text/css" media="print, handheld" href="example.css">
  </head>

<body>
   <p>The body...</p>
</body>
</html>
```

- The line links a CSS file that works only when the page is printed or viewed on handheld devices.
- The media attribute is used to apply a CSS file only for specific devices like print or handheld screens.

Recognized Media Types:

The names chosen for CSS media types reflect target devices for which the relevant properties make sense. They give a sense of what device the media type is meant to refer to. Given below is a list of various media types.

S.No.	Value & Description
1	**all:** Suitable for all devices.
2	**aural:** Intended for speech synthesizers.
3	**braille:** Intended for braille tactile feedback devices.
4	**embossed:** Intended for paged braille printers.
5	**handheld:** Intended for handheld devices (typically small screen, monochrome, limited bandwidth).
6	**print:** Intended for paged, opaque material and for documents viewed on screen in print preview mode. Please consult the section on paged media.
7	**projection:** Intended for projected presentations, for example projectors or print to transparencies. Please consult the section on paged media.
8	**screen:** Intended primarily for color computer screens.
9	**tty:** Intended for media using a fixed-pitch character grid, such as teletypes, terminals, or portable devices with limited display capabilities.
10	**tv:** Intended for television-type devices.

Media Queries

- **Media queries can be used to check many things, such as:**
    - width and height of the viewport.
    - orientation of the viewport (landscape or portrait).
    - Resolution.
- Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

- **CSS Media Types:**

Value	Description
All	Used for all media type devices
Print	Used for print preview mode
screen	Used for computer screens, tablets, smart-phones, etc.

- **CSS Common Media Features:** Here are some commonly used media features.

Value	Description
orientation	Orientation of the viewport (landscape or portrait)
Max-height	Maximum height of the viewport
Min-height	Minimum height of the viewport
Height	Height of the viewport (including scrollbar)
Max-width	Maximum width of the viewport
Min-width	Minimum width of the viewport
width	Width of the viewport (including scrollbar)

Example-1:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: orange;
}
/* Media Query */
@media screen and (min-width:
480px) {
  body {
    background-color: lightgreen;
  }
}
</style>
</head>
```

```
<body>
<h1>Resize the browser window to see the effect!!!</h1>
<p>
The media query will only apply if the media type is screen
and the viewport is 480px wide or wider.
</p>
</body>
```

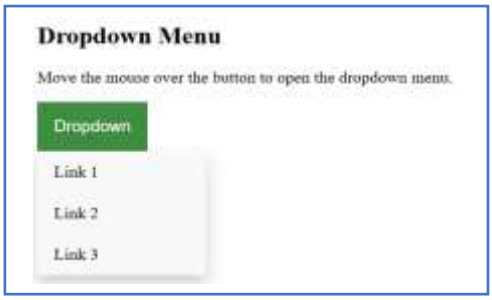
Drop-Down Menus

- Create a dropdown menu that allows the user to choose an option from a list:

```
<!DOCTYPE html>
<html>
<head>
<style>
.dropbtn {
  background-color: #4CAF50;
  color: white;
  padding: 16px;
  font-size: 16px;
  border: none;
  cursor: pointer;
}

.dropdown {
  position: relative;
  display: inline-block;
}

.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  z-index: 1;
}
```

```
.dropdown-content a {
  color: black;
  padding: 12px 16px;
  text-decoration: none;
  display: block;
}

.dropdown-content a:hover {
  background-color: #f1f1f1;
}

.dropdown:hover .dropdown-content {
  display: block;
}

.dropdown:hover .dropbtn {
  background-color: #3e8e41;
}
</style>
</head>
```

```
<body>

<h2>Dropdown Menu</h2>
<p>Move the mouse over the button to open the dropdown menu.</p>

<div class="dropdown">
  <button class="dropbtn">Dropdown</button>
  <div class="dropdown-content">
    <a href="#">Link 1</a>
    <a href="#">Link 2</a>
    <a href="#">Link 3</a>
  </div>
</div>

</body>
</html>
```


Dropdown Menu
Move the mouse over the button to open the dropdown menu.
Dropdown
Link 1
Link 2
Link 3

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

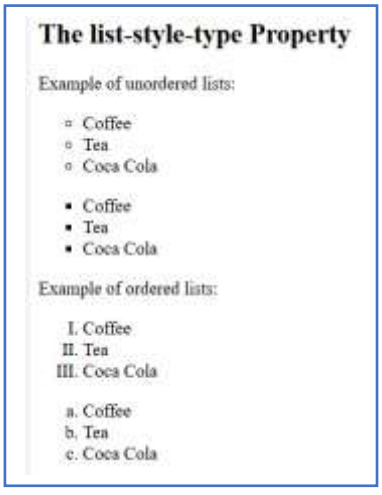
letters The CSS list properties allow you to:

- 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.

```
<!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>
```



Introduction to Cascading Style Sheet (CSS): Part 2

Text Shadows

The text-shadow property adds shadow to text.

This property accepts a comma-separated list of shadows to be applied to the text.

- **Syntax is:**

  text-shadow: h-shadow v-shadow blur-radius color | none | initial | inherit;

  - **h-shadow** → Horizontal shadow (left/right)
  - **v-shadow** → Vertical shadow (up/down)
  - **blur-radius** → Blur effect (optional)
  - **color** → Shadow color

Note: To add more than one shadow to the text, add a comma-separated list of shadows.

Property Values:

Value	Description
h-shadow	Required. The position of the horizontal shadow. Negative values are allowed.
v-shadow	Required. The position of the vertical shadow. Negative values are allowed.
blur- radius	Optional. The blur radius. Default value is 0.
color	Optional. The color of the shadow. Look at CSS Color Values for a complete list of possible color values.
none	Default value. No shadow.
initial	Sets this property to its default value.
inherit	Inherits this property from its parent element.

Example-1:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
        text-shadow: 2px 2px;
  }
</style>
</head>
```

```
<body>

<h1>The text-shadow Property</h1>

</body>
</html>
```

Example-2:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
        text-shadow: 2px 2px blue;
  }
</style>
</head>
```

```
<body>

<h1>The text-shadow Property</h1>

</body>
</html>
```

The text-shadow Property

- **Example-3: Text-shadow with a blur effect.**

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
        text-shadow: 2px 2px 8px green;
  }
</style>
</head>
```

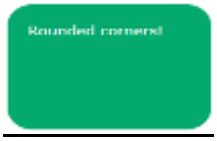
```
<body>

<h1>The text-shadow with blur effect</h1>

</body>
</html>
```

The text-shadow with blur effect

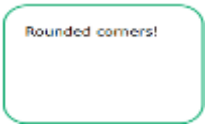
Rounded Corners

- The CSS border-radius property defines the radius of an element's corners.
- This property allows you to add rounded corners to elements!

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



2. Rounded corners for an element with a border:



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

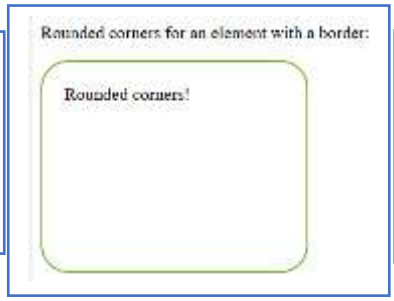
Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
#rcorners1 {
    border-radius: 25px;
    background: blue;
    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(nature.jpg);
    background-position: left top;
    background-repeat: repeat;
    padding: 20px;
    width: 200px;
    height: 150px;
}
</style>
</head>
```

```
<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>
```



Box Shadows

- The CSS box-shadow property is used to apply one or more shadows to an element.
- The Syntax is :

  box-shadow: none | h-offset v-offset blur spread color | inset | initial | inherit;

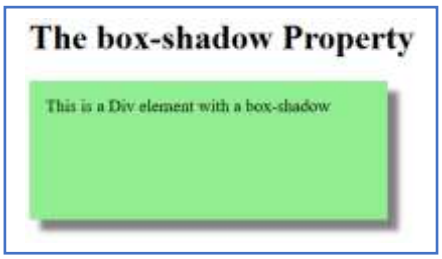
Value	Description
none	Default value. No shadow is displayed.
h-offset	Required. Horizontal offset of the shadow. Positive → right side, Negative → left side.
v-offset	Required. Vertical offset of the shadow. Positive → below the box, Negative → above the box.
blur	Optional. Blur radius. Higher value makes the shadow more blurred.
spread	Optional. Spread radius. Positive increases size, negative decreases size of the shadow.
color	Optional. Color of the shadow. Default is text color. In Safari (PC), color is required.
inset	Optional. Changes the shadow from outer (outset) to inner shadow.
initial	Sets the property to its default value.
inherit	Inherits the property from its parent element.

Example-1: Specify a Horizontal and a Vertical Shadow.

In its simplest use, you only specify a horizontal and a vertical shadow. The default color of the shadow is the current text-color.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   width: 300px;
   height: 100px;
   padding: 15px;
   background-color: lightgreen;
   box-shadow: 10px 10px 5px gray;
}
</style>
</head>
```

```
<body>
<h1>The box-shadow Property</h1>
<div>This is a Div element with a box-shadow</div>
</body>
</html>
```



Example-2: Increasing the Padding value.

Try the example by changing the padding value to 50px, 100px…;

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)

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 is:**

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

Example 1: Direction - Top to Bottom (this is default)

- The following example shows a linear gradient that starts at the top.

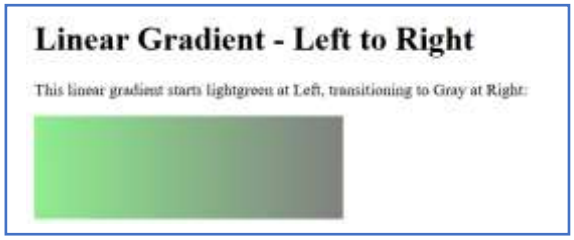
```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
   height: 100px;
   background-color: lightgreen;
   background-image: linear-
gradient(lightgreen, gray);
}
```

```
</style>
</head>

<body>
<h1>Linear Gradient - Top to Bottom</h1>
<p>This linear gradient starts lightgreen at the
Top, transitioning to Gray at the Bottom:</p>
<div id="grad1"></div>
</body>
</html>
```

- **Example 2: Direction - Left to Right**

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
  height: 100px;
  width: 300px;
  background-color: lightgreen;
  background-image: linear-gradient
(to right, lightgreen, gray);
}
</style>
</head>
```

```
<body>
<h1>Linear Gradient - Left to Right</h1>
<p>This linear gradient starts lightgreen at Left,
transitioning to Gray at Right:</p>
<div id="grad1"></div>
</body>
</html>
```



Example-3: 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).

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
  height: 100px;
  width: 300px;
  background-color: lightgreen;
  background-image: linear-gradient
(to bottom right, lightgreen, gray);
}
</style>
</head>
```

```
<body>
<h1>Linear Gradient - Diagonal</h1>
<p>This linear gradient starts lightgreen at Top
left, transitioning to Gray at Bottom-Right:</p>
<div id="grad1"></div>
</body>
</html>
```

Example 4: Direction - Diagonal

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
   height: 100px;
   width: 300px;
   background-color: lightgreen;
   background-image: linear-gradient
(to top right, lightgreen, gray);
}
</style>
</head>
```

```
<body>
<h1>Linear Gradient - Diagonal</h1>
<p>This linear gradient starts lightgreen at Bottom
left, transitioning to Gray at Bottom-Right:</p>
<div id="grad1"></div>
</body>
</html>
```

Radial Gradients

- A radial gradient is defined by its center.
- To create a radial gradient you must also define at least two color stops.

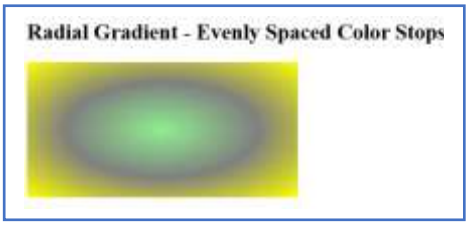
Syntax is :

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

Example-1:

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
   height: 150px;
   width: 300px;
   background-color: lightgreen;
   background-image: radial-gradient
(lightgreen, gray, yellow);
}
</style>
</head>
```

```
<body>
<h2>Radial Gradient - Evenly Spaced Color Stops</h2>
<div id="grad1"></div>
</body>
</html>
```



Example-2:

```
<!DOCTYPE html>
<html>
<head>
<style>
#grad1 {
   height: 150px;
   width: 200px;
   background-color: lightgreen;
   background-image: radial-gradient
(red, yellow, green);
}
</style>
</head>
```

```
<body>
<h2>Radial Gradient - Evenly Spaced Color Stops</h2>
<div id="grad1"></div>
</body>
</html>
```

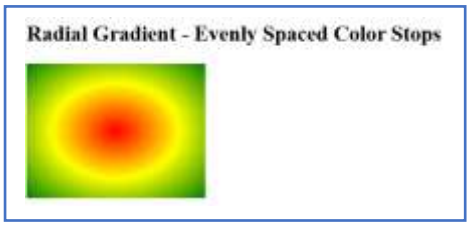


Image Borders

- **This CSS property defines an image to be used as the element's border. It draws an image outside the element and replaces the element's border with the corresponding image. It is an interesting task to replace the border of an element with the image.**
- It is the shorthand property for border-image-source, border-image-slice, border-image-width, border-image-outset, and border-image-repeat. We can set all these properties at once using the border-image property. If any of the values are not specified, then they set to their default values.

Syntax is:

border-image: source slice width outset repeat | initial | inherit;

border-image-source: It specifies the source of the border-image. It sets the path of the image, or we can say that it specifies the location of the image to be used as the border.

border-image-slice: It is used to divide or slice the image, which is specified by the border-image-source property.

The values of this property specify how to slice the image for creating the pieces of the border.

This property divides the image into nine sections that are:

- Four corners
- Four sides
- One center region

It can accept **four unitless positive values**. Its default value is **100%**.

border-image-width:

- It sets the width of the border-image.
- It can accept a unitless positive value, a percentage value, or the keyword **auto**. Its default value is **1**.
- We can specify up to four values for providing the width of individual sides.

border-image-outset: It sets the amount of space by which the border image is set out from its border box.

border-image-repeat: It controls the repetition of the image to fill the area of the border.

We can specify up to two values for this property.

- If we specify **one value**, then it is applied on both vertical and horizontal sides.

- If we specify **two values**, then:
    - the first value is applied on **horizontal sides**
    - the second value is applied on **vertical sides**

The values of this property are:

- stretch
- repeat
- round
- space

The default value of this property is **stretch**.

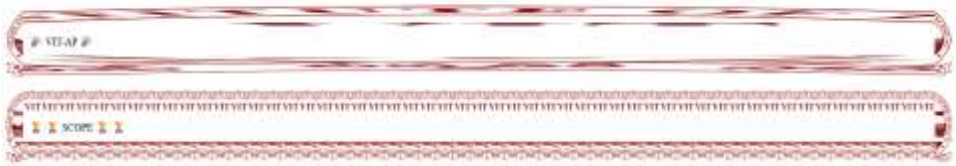
Initial: It sets the property to its default value. (**border-image: none 100% 1 0 stretch**).

Inherit: It inherits the property from its parent element.

Example:

```
<!DOCTYPE html>
<html>
<head>
<style>
#borderimg {
   border: 30px solid transparent;
   padding: 10px;
   border-image: url(VIT-AP_University_seal.png) 80 80 80 80 stretch;
}

#borderimg1 {
   border: 30px solid transparent;
   padding: 10px;
   border-image: url(VIT-AP_University_seal.png) 34% 34% repeat;
}
</style>
</head>

<body>

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

<p id="borderimg"> 🎉 VIT-AP 🎉 </p>

<p id="borderimg1"> 🏆 🏆 SCOPE 🏆 🏆 </p>

<p>Here is the original image:</p>
<img src="VIT-AP_University_seal.png">

</body>
</html>
```

The border-image Property



Here is the original image:

Animation

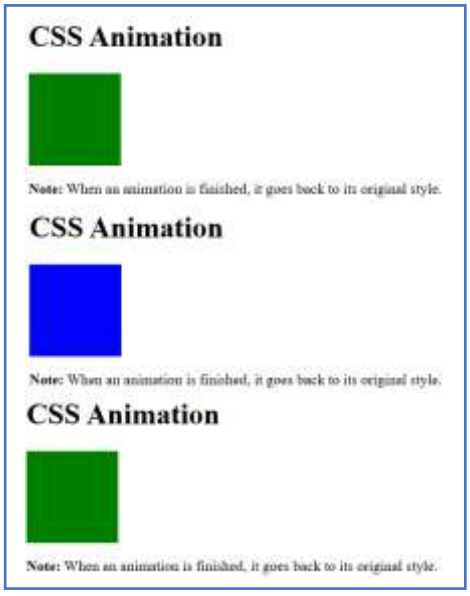
- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.

Property	Description
@keyframes	The @keyframes rule in CSS is used to specify the animation rule.
animation-name	It is used to specify the name of the @keyframes describing the animation.
animation-duration	It is used to specify the time duration it takes animation to complete one cycle.
animation-timing-function	It specifies how animations make transitions through keyframes. There are several presets available in CSS which are used as the value for the animation-timing-function like **linear**, **ease,ease-in**,**ease-out**, and **ease-in-out.**
animation-delay	It specifies the delay of the start of an animation.
animation-iteration-count	This specifies the number of times the animation will be repeated.
animation-direction	It defines the direction of the animation. animation direction can be **normal**, **reverse**, **alternate**, and **alternate-reverse**.
animation-fill-mode	It defines how styles are applied before and after animation. The animation fill mode can be none, forwards, backwards, or both.
animation-play-state	This property specifies whether the animation is running or paused.

Example-1:

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

@keyframes example {
  from { background-color: green; }
  to { background-color: blue; }
}
</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>
```



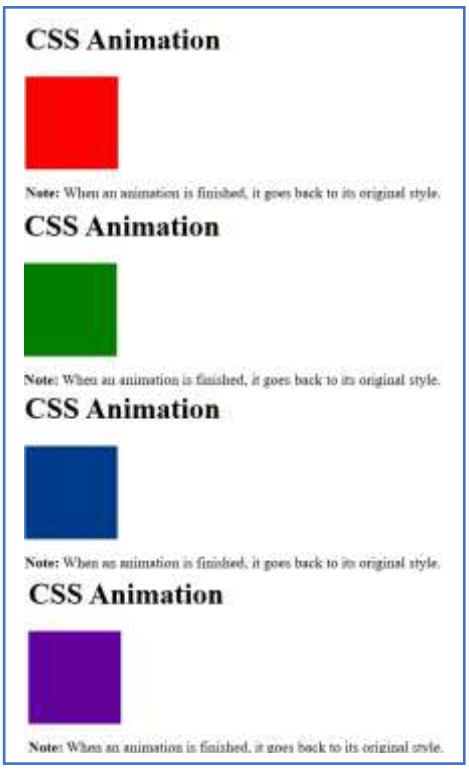
Example-2:

- **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).
- 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 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: green; }
  50%  { background-color: blue; }
  100% { background-color: purple; }
}
</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>
```



Example-3:

- 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:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-delay: 2s;
}

@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 following example has a 2 seconds delay before starting
the animation:</p>
<div></div>

</body>
</html>
```

Example-4:

- 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:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 4s;
  animation-iteration-count: 3;
}

@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 following example will run the
animation 3 times before it stops:</p>
<div></div>

</body>
</html>
```

Example-5:

- 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.

```
<!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 following example will run the animation in reverse direction (backwards):</p>
<div></div>

</body>
</html>
```

Example-6:

- The example below uses six of the animation properties.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background-color: red;
  position: relative;
  animation-name: example;
  animation-duration: 5s;
  animation-timing-function: linear;
  animation-delay: 2s;
  animation-iteration-count: infinite;
  animation-direction: alternate;
}
```

```
@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>This example uses six of the animation properties:</p>
<div></div>

</body>
</html>
```

Transitions

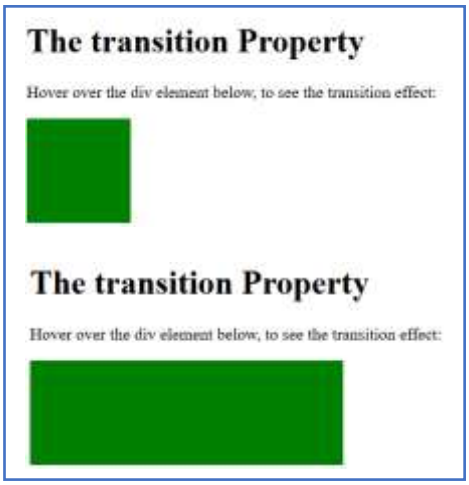
- **CSS transitions allow you to change property values smoothly, over a given duration.**
- CSS transitions are used to create smooth animations between two states of an element, enhancing interactivity and user experience.
- Transitions can animate properties like color, size, and position.
- The following table lists all the CSS transition properties:

Value	Description
transition	A shorthand property for setting all four transition properties into a single property
transition-delay	Specifies a delay (in seconds) for the transition effect
transition-duration	Specifies how many seconds or milliseconds a transition effect takes to complete
transition-property	Specifies the name of the CSS property the transition effect is applied to
transition-timing-function	Specifies the speed curve of the transition effect

- **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.
- **Example 1: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:**

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   width: 100px;
   height: 100px;
   background: green;
   transition: width 2s;
}

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

```
<body>
<h1>The transition Property</h1>
<p>Hover over the div element below, to see the transition effect:</p>
<div></div>
</body>
</html>
```

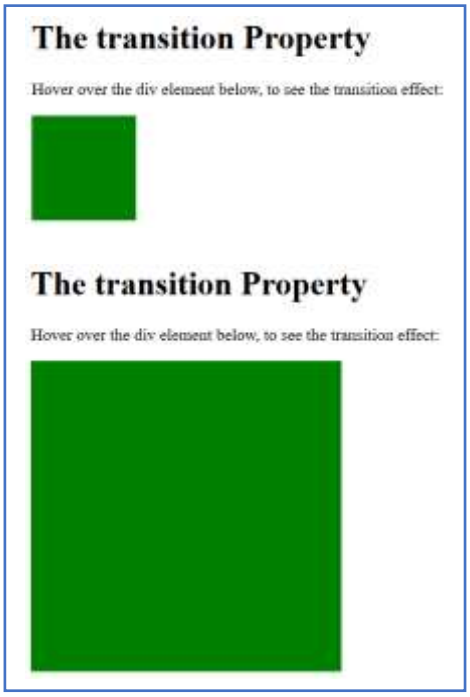


Example-2: Change Several Property Values

The following example adds a transition effect for both the width and height property, with a duration of 2 seconds for the width and 4 seconds for the height.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   width: 100px;
   height: 100px;
   background: green;
   transition: width 2s,
height 4s;
}

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

```
<body>
<h1>The transition Property</h1>
<p>Hover over the div element below, to see
the transition effect:</p>
<div></div>
</body>
</html>
```



Example-3: Delay the Transition Effect:

- The transition-delay property specifies a delay (in seconds) for the transition effect.
- The following example has a 2 second delay before starting:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
   width: 100px;
   height: 100px;
   background: green;
   transition: width 2s;
   transition-delay: 2s;
}

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

```
<body>
<h1>The transition-delay Property</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 2
second delay before starting:</p>
</body>
</html>
```

Example-4: Specify the Speed Curve of the Transition

The transition-timing-function property specifies the speed curve of the transition effect. The transition-timing-function property can have the following values:

- ease - specifies a transition effect with a slow start, then fast, then end slowly (this is default).
- linear - specifies a transition effect with the same speed from start to end.
- ease-in - specifies a transition effect with a slow start.
- ease-out - specifies a transition effect with a slow end.
- ease-in-out - specifies a transition effect with a slow start and end.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div {
      width: 100px;
      height: 100px;
      background: skyblue;
      transition: width 2s;
    }

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

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

  <h1>The transition-timing-function Property</h1>
  <p>Hover over the div elements below, to see the different speed curves:</p>

  <div id="div1">linear</div><br>
  <div id="div2">ease</div><br>
  <div id="div3">ease-in</div><br>
  <div id="div4">ease-out</div><br>
  <div id="div5">ease-in-out</div><br>

</body>
</html>
```

Example-5: The CSS transition properties can be specified one by one, like this.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div {
      width: 100px;
      height: 100px;
      background: red;
      transition-property: width;
      transition-duration: 2s;
      transition-timing-function: linear;
      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-6: using the shorthand property transition

Shorthand Property: You can combine all four transition properties into a single shorthand property, which simplifies the code and ensures readability.

Syntax

transition: (property name) | (duration) | (timing function) | (delay);

The transition property is a shorthand that allows you to define all four transition-related properties in one line:

- **property name** – the name of the CSS property to apply the transition to (e.g., width, background-color, all)
- **duration** – how long the transition should take (e.g., 2s, 500ms)
- **timing function** – the speed curve of the transition (e.g., ease, linear, ease-in-out, cubic-bezier(...))
- **delay** – how long to wait before starting the transition (e.g., 1s, 0.5s)

Transformations

- **A transformation in CSS is used to modify an element by its shape, size, and position. It transforms the elements along the X-axis and Y-axis.**
- The transform property applies a 2D or 3D transformation to an element. This property allows you to rotate, scale, move, skew, etc., elements.
- The transform CSS property lets you rotate, scale, skew, or translate an element. It modifies the coordinate space of the CSS visual formatting model.

- **Syntax :**
  transform: none | transform-functions| initial | inherit;
- There are 6 main types of transformation which are listed below:
  - translate()
  - rotate()
  - scale()
  - skewX()
  - skewY()
  - matrix()

translate() Method: The translate() method is used to move the element from its actual position along the X-axis and Y-axis.

rotate() Method: The rotate() method rotates an element clockwise or counter-clockwise.

Counter-clockwise rotation: Use negative values to rotate the element counter clockwise.

scale() Method: It is used to increase or decrease the size of an element according to the parameter given for the width and height.

skewX() Method: his method is used to skew an element in the given angle along the X-axis.

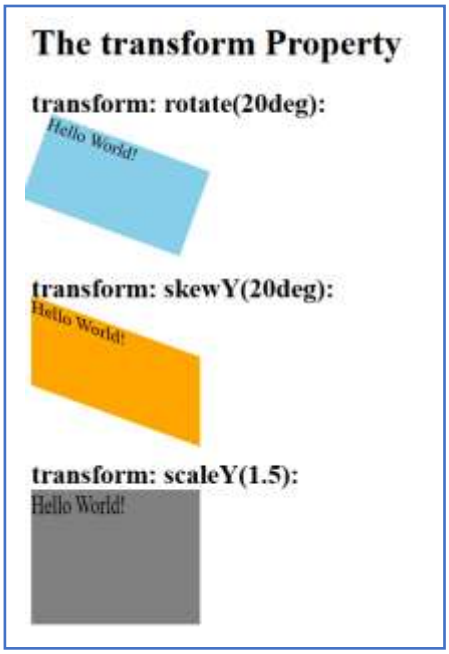
skewY() Method: This method is used to skew an element in the given angle along the Y-axis.

skew() Method: This method skews an element in the given angle along the X-axis and the Y-axis. The following example skews the element 20 degrees along the X- axis and 10 degrees along the Y-axis.

matrix() Method: This method combines all the 2D transform property into a single property. The matrix transform property accepts six parameters as matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY()).

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div.a {
      width: 150px;
      height: 80px;
      background-color: skyblue;
      transform: rotate(20deg);
    }

    div.b {
      width: 150px;
      height: 80px;
      background-color: orange;
      transform: skewY(20deg);
    }

    div.c {
      width: 150px;
      height: 80px;
      background-color: gray;
      transform: scaleY(1.5);
    }
  </style>
</head>
```

```
<body>

  <h1>The transform Property</h1>

  <h2>transform: rotate(20deg):</h2>
  <div class="a">Hello World!</div>
  <br>

  <h2>transform: skewY(20deg):</h2>
  <div class="b">Hello World!</div>
  <br>

  <h2>transform: scaleY(1.5):</h2>
  <div class="c">Hello World!</div>

</body>
</html>
```



Skew

The skew() function is an inbuilt function which is used to transform an element in the 2D plane. Skew an element means to pick a point and push or pull it in different directions.

Syntax: *Skew(ax) or skew(ax, ay)*

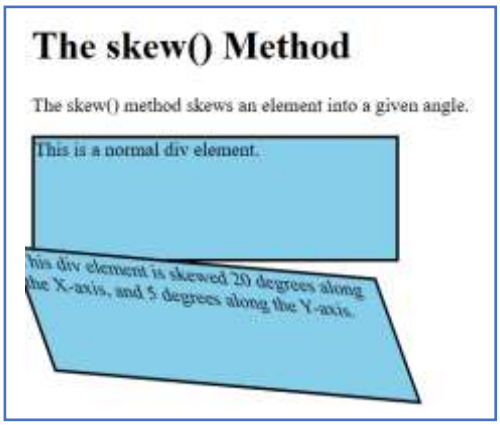
Parameters:

- **ax:** This parameter holds the angle representing the horizontal axis to distort an element.
- **ay:** This parameter holds the angle representing the vertical axis to distort an element. If it is not defined then it takes the default value zero. It means completely skew in x direction.

Example 1 : The element is skewed 20 degrees along the X-axis, and 5 degrees along the Y-axis.

```
<!DOCTYPE html>
<html>
<head>
  <style>
    div {
      width: 300px;
      height: 100px;
      background-color: skyblue;
      border: 3px solid black;
    }

    div#myDiv {
      transform: skew(20deg, 5deg);
    }
  </style>
</head>
```

```
<body>

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

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

  <div id="myDiv">
    This div element is skewed 20 degrees along the X-axis, and 5 degrees along the Y-axis.
  </div>

</body>
```



Example-2: The element is skewed 20 degrees along the X-axis, and 10 degrees along the Y-axis.

Example-3: The element is skewed 20 degrees along the X-axis, and 20 degrees along the Y-axis.

Example-4: The element is skewed 20 degrees along the X-axis, and 30 degrees along the Y-axis.

Transitioning Between Images

- **We can also use the transition property to create the visually beautiful effect of melting one image into another.**
- The transition property includes three values.
- First, we specify that the transition will occur on the opacity of the image.
- The second value is the transition-duration.
- The third value(ease-in-out) is the transition timing-function