**CSS Introduction**

CSS (Cascading Style Sheets) is a styling language used to add style to a webpage.

HTML provides structure and adds content to a webpage, while CSS enhances the visual presentation of that content through various styles.

selector {

property1: value;

property2: value;

}

The basic syntax of CSS includes **3** main parts:

* selector - specifies the HTML element that we want to apply the styles
* property1 **/** property2- specifies the attribute of HTML elements that we want to change (color, background, and so on)
* value - specifies the new value you want to assign to the property (color of the text to red, background to gray, and so on)

**Include CSS in a webpage**

CSS is used for styling the look and formatting of a document written in HTML. There are three ways to add CSS in HTML

* **Inline CSS**: Styles added directly to the HTML element.
* **Internal CSS**: Styles defined at the head section of the document.
* **External CSS**: Styles defined in a separate file.

we have CSS in a separate file named **style.css**. The external CSS file should have a **.css** extension.

<head>

<link href="style.css" rel="stylesheet">

</head>

**CSS Selectors**

CSS selectors are used to select the HTML elements that are to be styled by CSS.

**Types of Selector**

There are the following different types of selectors in CSS.

* Element selector

The element selector selects HTML elements (p, div, h1, etc) and applies CSS to them

* Id selector
* The id selector selects the HTML element with a unique identifier (id) and adds CSS to it.
* The id selector is specified using the hash (#) character, followed by the id of the element.

**Note**: The id selector is unique and selects one unique element.

* Class selector
* The class selector selects the HTML element using the class attribute and applies CSS to it.
* The class selector is specified using the period (.) character, followed by the class name.
* Universal selector

The universal selector selects every single HTML element on the page. It is written using the asterisk ( \* ) character.

**Note**: The universal selector is also referred to as the wildcard selector.

The style rule hierarchy determines the priority of CSS rules when multiple rules target the same element.

In CSS, the following hierarchy of style priorities applies:

* Inline styling: Styles applied directly within HTML element.
* ID selectors: Styles elements with specific ID.
* Class and attribute selectors: Styles elements with certain class or attribute.
* Element selectors: Styles elements with specific tag name

*id selector has more priority than the class and element selectors.*

**CSS Background**

CSS background property is used to add a color or an image to the background of an element.

* [background-image](https://www.programiz.com/css/background-image): allows to add an image as a background of an element
* [background-size](https://www.programiz.com/css/background-size): specifies the size of the background image
* [background-repeat](https://www.programiz.com/css/background-repeat): controls the repeating behavior of a background image
* [background-attachment](https://www.programiz.com/css/background-attachment): controls whether the background image scrolls or remains fixed with the page's content
* [background-clip](https://www.programiz.com/css/background-clip): defines the background area for an element to clip
* [background-color](https://www.programiz.com/css/background-color): sets the background color in an element

**CSS Font**

CSS font properties are used to adjust the appearance of the text in an HTML document. Using the CSS fonts properties, we can customize the font family, size, weight, style, and color of text.

* font-family: defines the font applied to the text
* font-size: sets the size of the font
* font-weight: sets the thickness i.e increase the boldness or lightness of the font
* font-style: sets the font to italic or oblique
* font-variant: changes the font to small-caps
* font-stretch: expands or narrows the text
* line-height: sets the distance between lines of the text

**CSS Text Formatting**

text-decoration: decorate the appearance of text using various lines

* text-decoration-line: sets the type of line decoration for our text
* text-decoration-color: sets the color for the line decoration
* text-decoration-style: sets the style of the line specified by text-decoration-line
* text-decoration-thickness: sets the thickness of the line used in decoration

p {

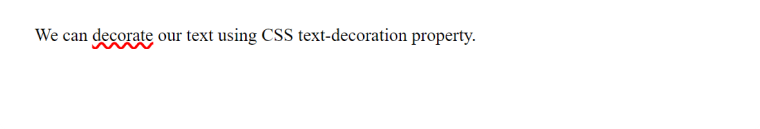
text-decoration-line: underline;

text-decoration-style: wavy;

text-decoration-color: red;

text-decoration-thickness: 2px;

}

****

**Text Spacing Properties**

* letter-spacing
* word-spacing
* text-indent
* line-height
* 1) letter-spacing

p.positive\_value {

letter-spacing: 2px;

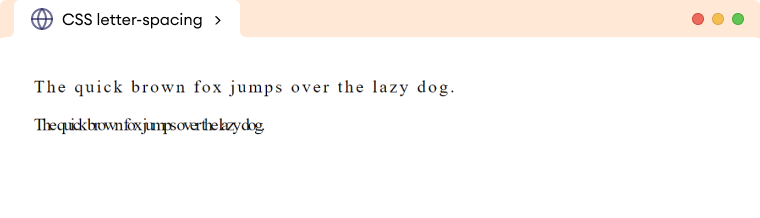
}

/\* narrows the space between characters by 2px \*/

p.negative\_value {

letter-spacing: -2px;

}

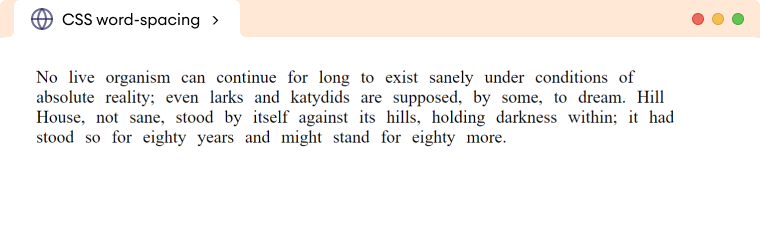
2) word-spacing

/\* widens the space between words by 6px \*/

p {

word-spacing: 6px;

}

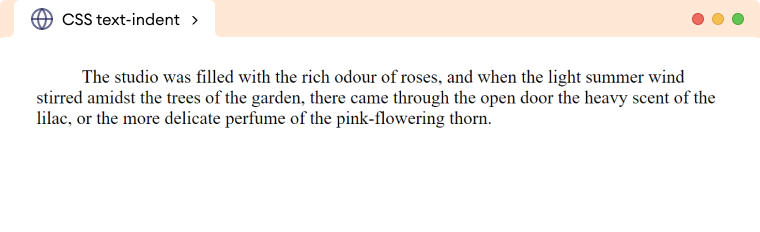


3) Text-Indent

p {

text-indent: 40px;

}

****

4) Line-Height

p.normal-value {

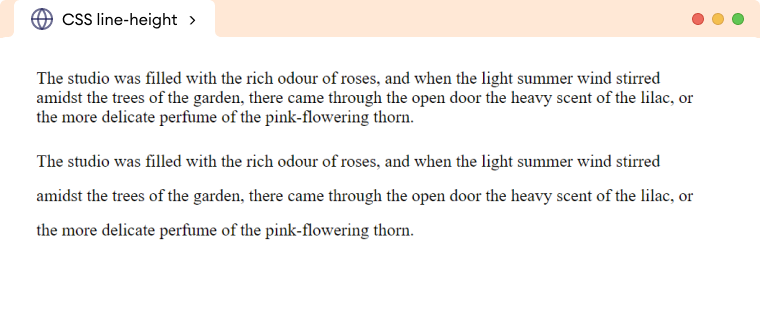
line-height: normal;

}

p.numeric-value {

line-height: 2;

}

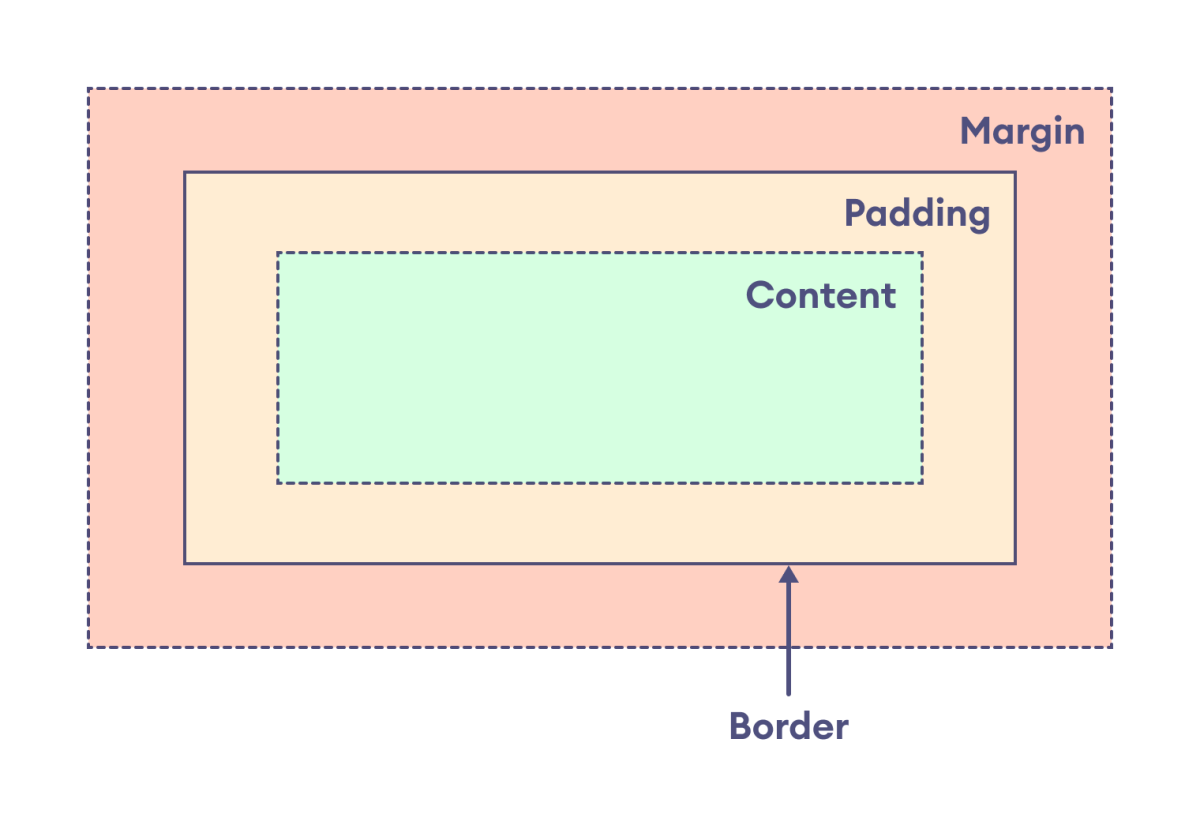


**CSS Box Model**

The CSS box model is a fundamental concept that defines how the element's dimensions and spacing are calculated.

The box model treats every HTML element as a rectangular box consisting of content, padding, border, and margin.

The box model defines the layout of an HTML element in the following way:



The components of the box model are:

* content: actual text or image that is displayed in the element
* padding: transparent space between the content and the border of an element
* border: line that surrounds the padding and content within the element
* margin: transparent area added outside the border

The primary purpose of the box model is to explain how the dimensions and spacing of elements are calculated and how they relate to each other.

## Width and Height of an Element with Box Model

The box model is important for understanding how the width and height of an element are calculated.

The width and height of the element are applied only to the content of the element by default. Hence, the actual size of the element is calculated by adding the padding and border along with the specified width and height of the element.

Actual width: border-left + padding-left + width + padding-right + border-right

Actual height: border-top + padding-top + height + padding-bottom + border-bottom

div {

width: 400px;

height: 80px;

border: 10px solid black;

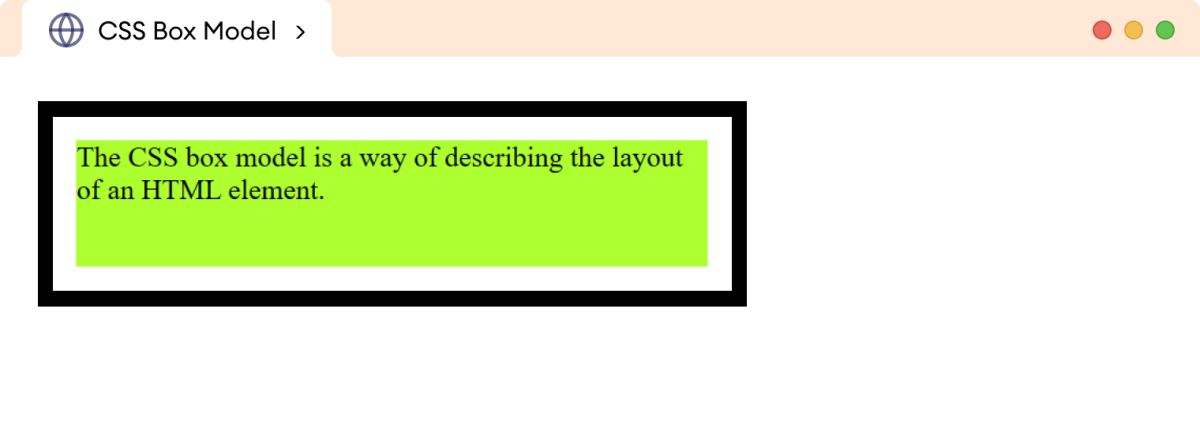
padding: 15px;

background-color: greenyellow;

/\* clips the background color to content only \*/

background-clip: content-box;

}

****

**Inline vs. Block Elements**

* **All HTML elements (tags) are assigned a display property value of either inline or block.**
* **Inline elements display in browsers horizontally.**

**[INLINE ELEMENT 1] [INLINE ELEMENT 2] [INLINE ELEMENT 3]**

* **Block elements display in browsers vertically (stacked one on top of the other).**

**[BLOCK ELEMENT 1]  
[BLOCK ELEMENT 2]  
[BLOCK ELEMENT 3]**

* **Examples of inline elements:**

**<a><img><strong><em><span>**

* **Examples of block elements:**

**<p><h1-h6><div><hr><table><ul><ol>**

**Using CSS, you can change this inherent display property:**

**To force a block display, use the declaration display: block;**

**To force an inline display, use the declaration display: inline;**

**To hide elements, use the declaration display: none;**

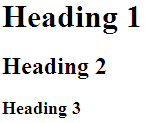
1. **Normally, <a> tags display inline.**

****

**But, if you add the style a {display: block;}, they will display as a vertical navigation menu:**

****

**2)Normally, the heading tags display in block format:**

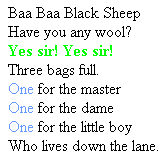
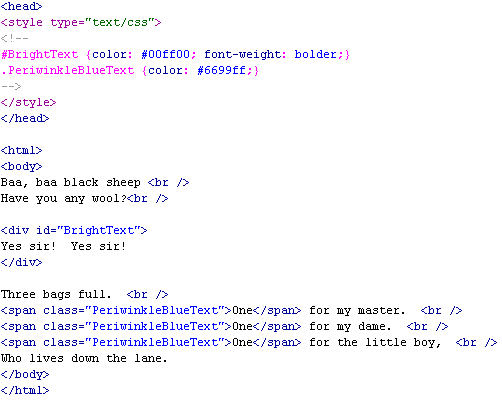
****

**But, to have them display inline, add the style h1,h2,h3{display: inline;}**

****

**Span and Div**

* **There are two tags that are particularly useful when using CSS: <span> and <div>. They are both container tags that have minimal formatting associated with them.**
* **The <span> tag is an inline element that simply holds text without doing anything special to it.**
* **The <div> tag is a block element and causes the text it encloses to start on a new line.**
* **Using <span> and <div> tags in conjunction with classes and IDs allows for great flexibility in creating pages.**

****

**TABLES**

table,th,td {

border: 1px solid black;

}

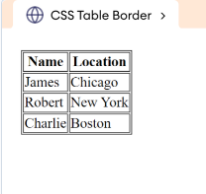
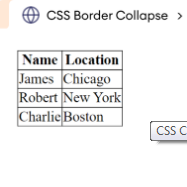
****

table {

border-collapse: collapse;

}

****

**Table Layout**

The table-layout property specifies the structure and behavior of the table.

It can take one of the following values:

* auto: adjusts the column widths automatically based on content
* fixed: specifies the fixed column width

table {

width: 500px;

table-layout: fixed;

}

table, th , td {

border: 1px solid black;

border-collapse: collapse;

}

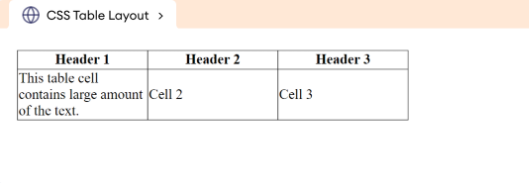
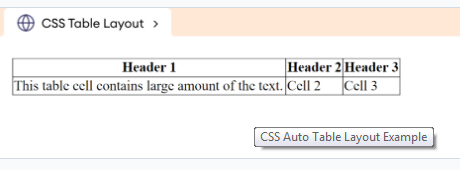
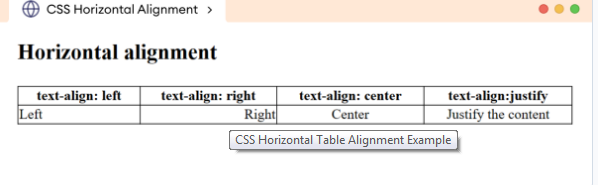
****

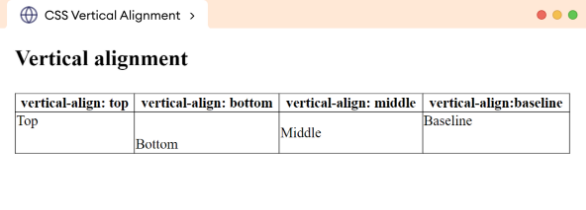
table-layout: auto;

****

**Text-align sets the horizontal alignment (like left, right, or center)**

****

**vertical-align sets the vertical alignment (like top, bottom, or middle)**

****

**Table Padding**

**th, td {  
  padding: 15px; //space between the border and the content in a table  
  text-align: left;  
}**

* **tr:hover {background-color: coral;} //to highlight table rows on mouse hover**
* **tr:nth-child(even) {background-color: #f2f2f2;} //Striped Tables**
* **th {  
    background-color: #04AA6D;  
    color: white;  
  }**

**List**

* list-style-type
* disc: specifies a filled circle
* circle: specifies a hollow circle
* square: specifies a filled square
* decimal: represents decimal numbers starting with 1
* lower-alpha: specifies lowercase ASCII letters
* lower-roman: specifies lowercase Roman numerals
* upper-alpha: specifies uppercase ASCII letters
* upper-roman: specifies uppercase Roman numerals
* list-style-position

.inside-list {

list-style-position: inside;

}

.outside-list {

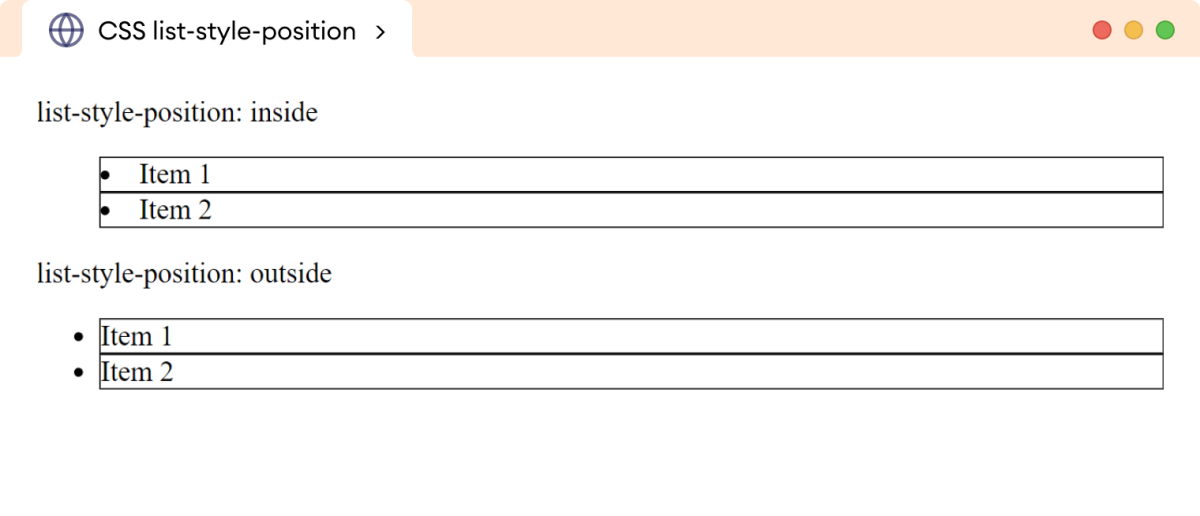
list-style-position: outside;

}

li {

border: 1px solid black;

}



* list-style-image

ul {

list-style-image: url("https://openclipart.org/image/20px/183195");

}



**Nesting in CSS**

Nesting in CSS allows you to nest one style rule inside another. The child rule’s selector is relative to the parent rule’s selector. This method enhances the modularity and maintainability of CSS stylesheets, making the code more readable.

<**style**>

**p** **a** {

**color**: **green**;

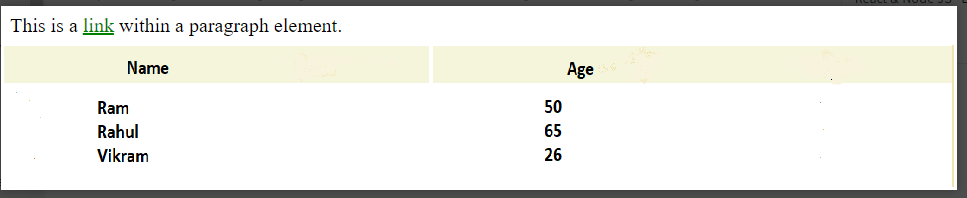
}

**table** **tr** **th** {

**background-color**: **beige**;

}

</**style**>

****

**Grouping in CSS**

Grouping allows you to apply common styling properties to multiple elements at once. This reduces the length of your code, makes it easier to read, and speeds up page load times.

Instead of writing this long code, specifying the same properties to different selectors:

h1 {

padding: 5px;

color: grey;

}

p {

padding: 5px;

color: grey;

}

h1, p {

padding: 5px;

color: grey;

}

# CSS Positioning Elements

CSS positioning allows you to control the layout of HTML elements using properties such as top, right, bottom, and left. Understanding the different types of position properties in CSS—static, relative, absolute, fixed, and sticky—can enhance the design and functionality of web pages.

| **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. |

1. **Static :** HTML elements are positioned static by default.

Static positioned elements are not affected by the top, bottom, left, and right properties.

Elements render in order, as they appear in the document flow

<html>

<head>

<style>

div.static {

position: static;

border: 3px solid #73AD21;

}

</style>

</head>

<body>

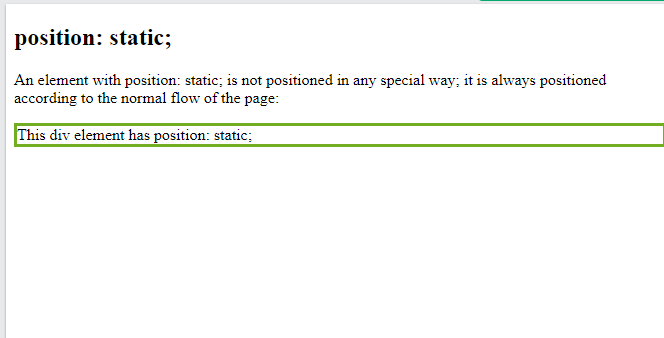
<div class="static">

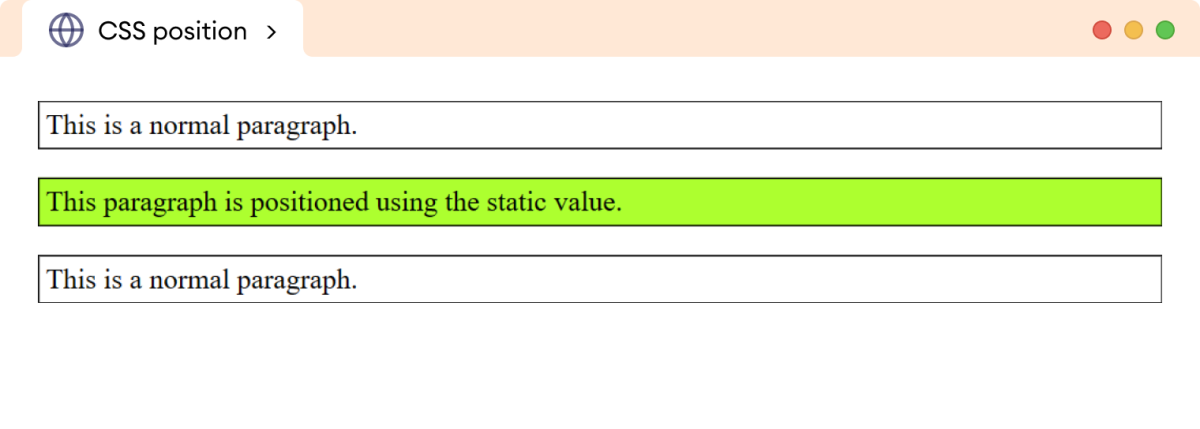
This div element has position: static;

</div>

</body>

</html>





1. **Relative :** Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position.

The element is positioned relative to its normal position, so "left:20px" adds 20 pixels to the element's LEFT position

<style>

div.relative {

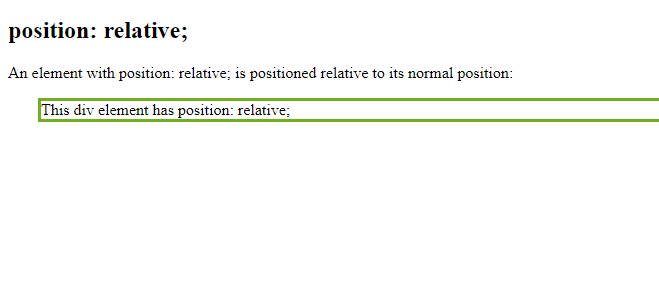
position: relative;

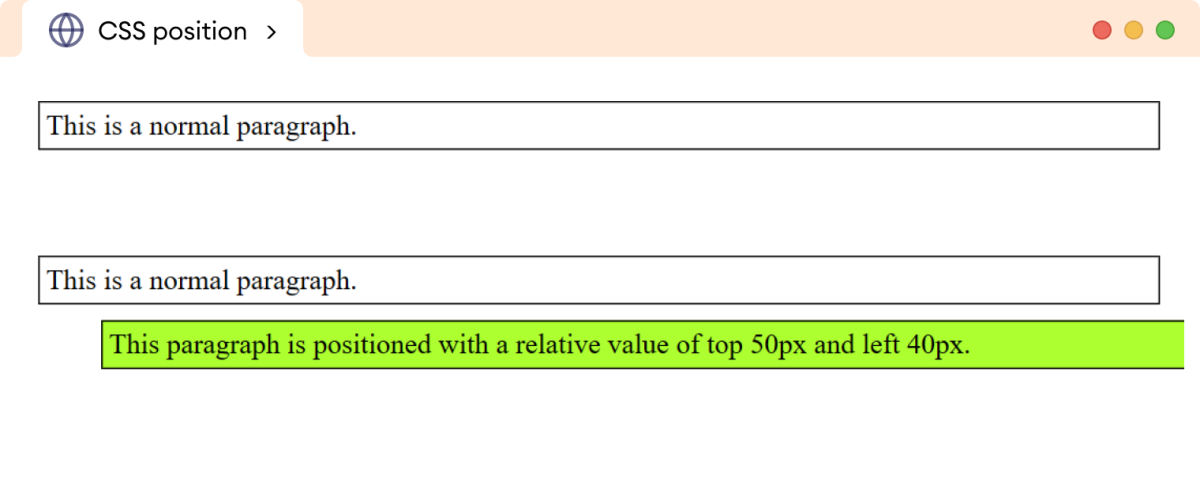
left: 30px;

border: 3px solid #73AD21;

}

</style>





## 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 element is positioned relative to the browser window

<style>

div.fixed {

position: fixed;

bottom: 0;

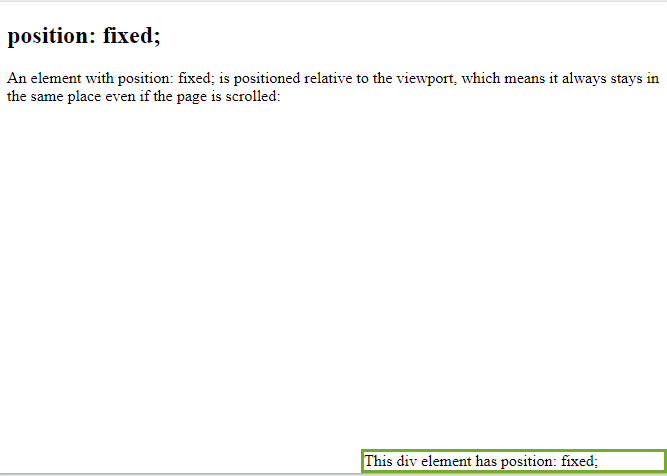
right: 0;

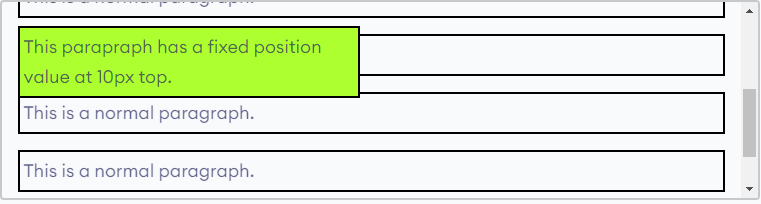
width: 300px;

border: 3px solid #73AD21;

}

</style>





## position: absolute;

An element with position: absolute; is positioned relative to the nearest positioned ancestor .

The absolute value removes the element completely from its normal flow in the document.

If there is no positioned ancestor, they are positioned relative to the document itself.

<style>

div.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

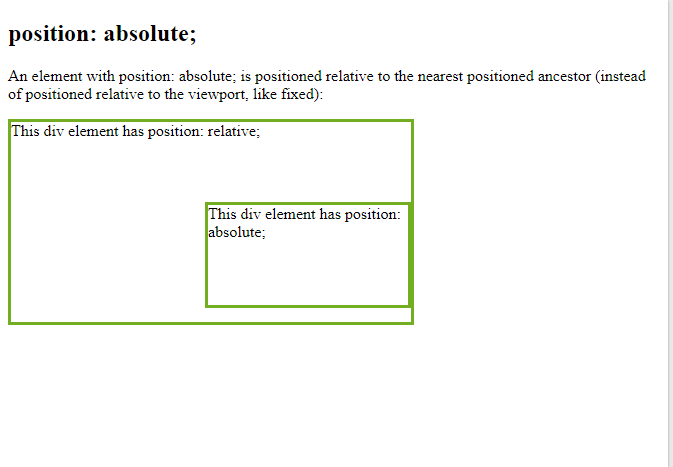
width: 200px;

height: 100px;

border: 3px solid #73AD21;

}

</style>



## position: sticky;

An element with position: sticky; is positioned based on the user's scroll position.

The sticky value positions the element as a combination of relative and fixed values.

The sticky position behaves like relative positioning until the element reaches a certain scroll point on the screen. After that, the element sticks to the top of the viewport like a fixed element.

<style>

div.sticky {

position: sticky;

top: 0;

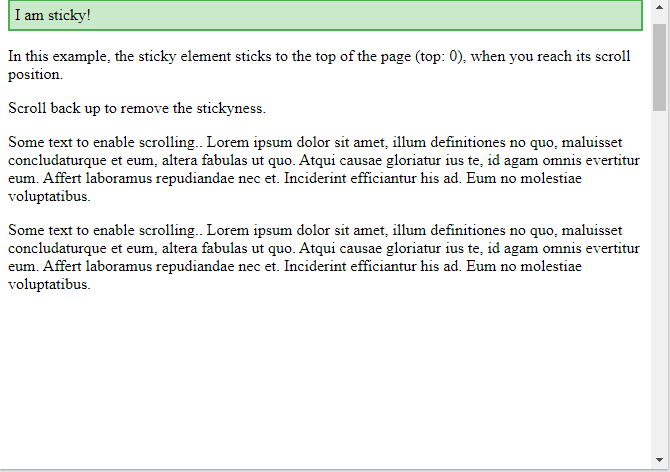
padding: 5px;

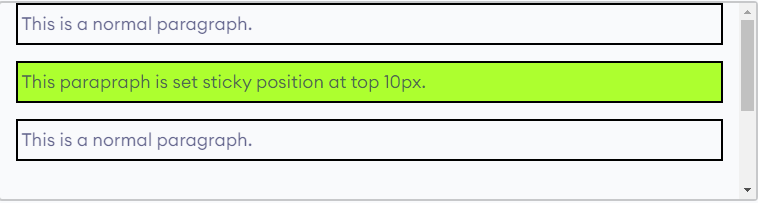
background-color: #cae8ca;

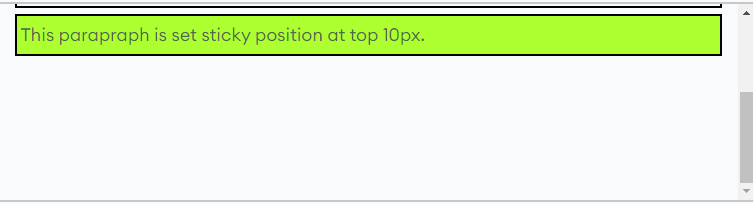
border: 2px solid #4CAF50;

}

</style>







What are Pseudo-classes?

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

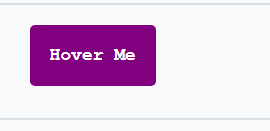
* Style an element when a user mouses over it
* Style visited and unvisited links differently
* Style an element when it gets focus

button:hover {

background-color: red;

color:black;

}





## Anchor Pseudo-classes

/\* 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;  
}

## Hover on <div>

An example of using the :hover pseudo-class on a <div> element:

div {

background-color: green;

color: white;

padding: 25px;

text-align: center;

}

div:hover {

background-color: blue;

}

What are Pseudo-Elements?

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

## The ::first-line Pseudo-element

The ::first-line pseudo-element is used to add a special style to the first line of a text.

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

## The ::first-letter Pseudo-element

The ::first-letter pseudo-element is used to add a special style to the first letter of a text.

<style>

p::first-letter {

color: #ff0000;

font-size: xx-large;

}

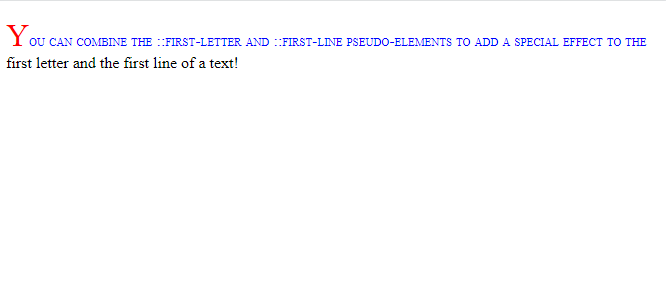
p::first-line {

color: #0000ff;

font-variant: small-caps;

}

</style>



## CSS - The ::before Pseudo-element

The ::before pseudo-element can be used to insert some content before the content of an element.

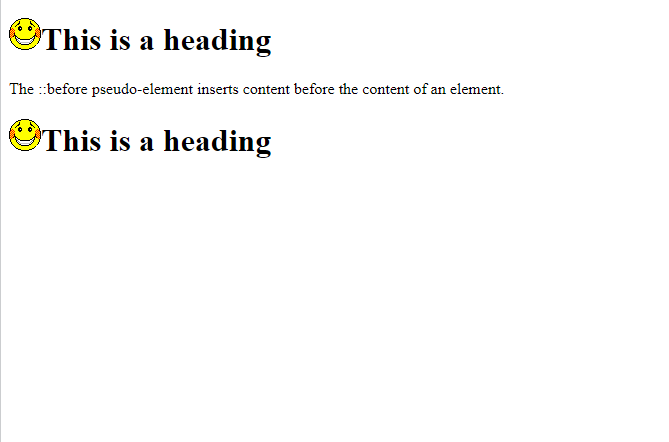
<style>

h1::before {

content: url(smiley.gif);

}

</style>



**CSS - Units**

CSS Units define the measurement system used to specify the values. CSS offers a number of different units for expressing length and measurement. CSS unit is used to specify the property size for a page element or its content.

* **Absolute units:**Fixed unit lengths that does not depend on screen width.
* **Relative units:**Responsive unit lengths that changes according to screen width

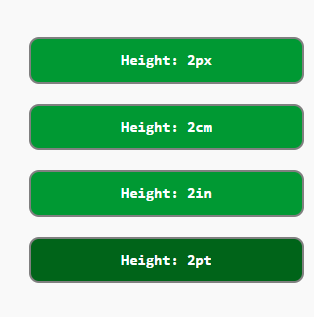
## Absolute Length Units

These units are categorized as fixed-length units, which means that lengths specified with absolute units maintain an exact, unchanged size on the screen.

 it is a fixed-size unit of measurement that does not change with resizing

mm,cm,in,pt,px

Absolute units prove valuable for projects where responsiveness is not a priority.



1pt = 1/72 of 1in

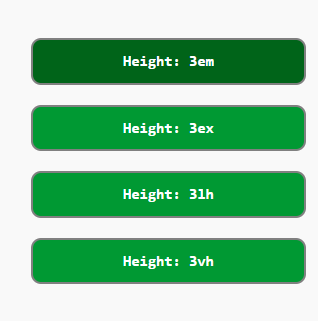
1px = 1/96th of 1in

## Relative Length Units

Relative length units are measured in relation to other elements or viewport of the screen.

Relative units are great for styling responsive websites because they can be adjusted proportionally based on window size or parent elements. These units define lengths relative to other length properties.

Em, ex, rem,em



The rem unit is only relative to the document's root element, while the em unit is only relative to the immediate parent of the targeted element.

**CSS Dimensions**

CSS dimension define 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.

## CSS Setting height and width

* **length**: Any unit of length (px, pt, em, in, etc.)
* **percentage (%)**: A percentage value, which is in percent height/width of the containing block.
* **auto**: Browser calculates the height and width of the element. (For example setting height automatically to match aspect ratio of image for the specified width)
* **initial**: Sets the value of height and width to its default value.
* **inherit**: Inherits the value of height and width from its parent value.

**<style>**

**div {**

**height: 100px;**

**width: 80%;**

**background-color: rgb(206, 211, 225);**

**}**

**img{**

**height: auto;**

**width: 180px;**

**}**

**</style>**

## Set Max-Height and Max-Width

The [**max-height**](https://www.tutorialspoint.com/css/css_max-height.htm) and [**max-width**](https://www.tutorialspoint.com/css/css_max-width.htm) properties are used to set the maximum height and width of an element.

* **max-width**: Sets the maximum width an element can be. Prevents an element from exceeding this width, even if the content inside it requires more space.
* **max-height**: Sets the maximum height an element can be. Prevents an element from exceeding this height, even if the content inside it requires more space.

## Set Min-Height and Min-Width

The [**min-height**](https://www.tutorialspoint.com/css/css_min-height.htm) and [**min-width**](https://www.tutorialspoint.com/css/css_min-width.htm) properties are used to set the minimum height and width of an element.

* **min-width**: Sets the minimum width an element can be. Ensures that the element doesn’t shrink below this width, even if the content is smaller.
* **min-height**: Sets the minimum height an element can be. Ensures that the element doesn’t shrink below this height, even if the content is smaller.

**CSS Display**

The display property is used to specify how an element is shown on a web page.

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

The display property is used to change the default display behavior of HTML elements.

The commonly used values for the display property are as follows:

* inline: allows an element to behave like an inline-level element
* block: allows an element to behave like a block-level element
* inline-block: formats the element as inline-level but also allows to set height/width like block-level element
* flex: sets the element as a flex container to have a flexible layout of its child elements
* grid: sets the element as a grid container to create complex layouts
* none: removes the element from the document leaving no space

**Inline:**

div {

border: 2px solid black;

margin-bottom: 12px;

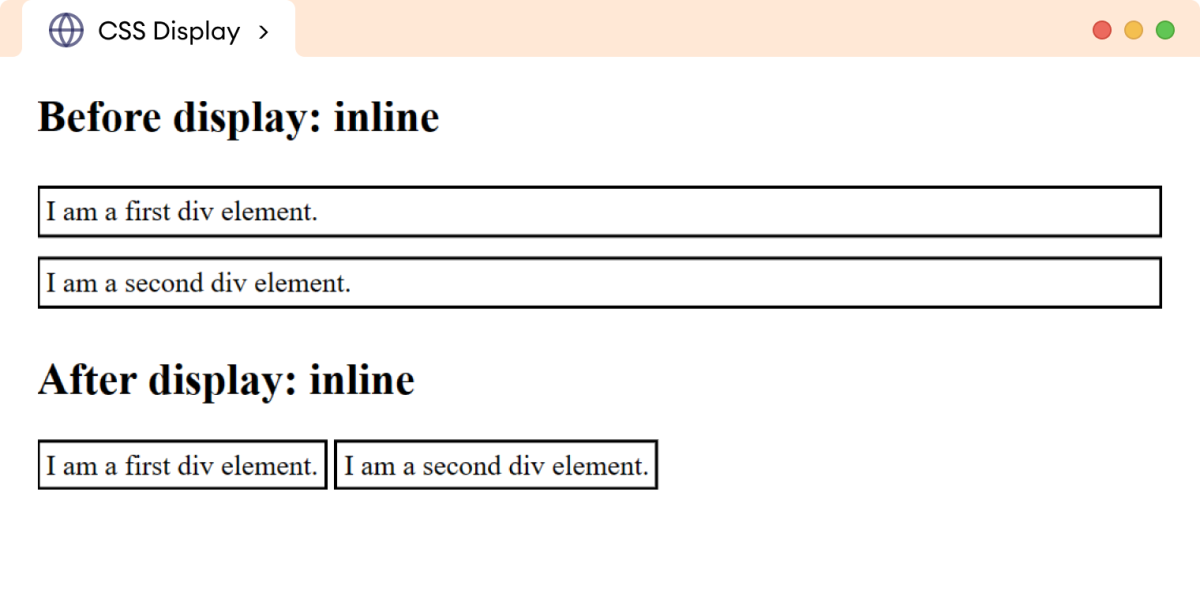
padding: 4px;

}

div.after {

display: inline;

}

****

## Block:

span {

border: 2px solid black;

margin-bottom: 12px;

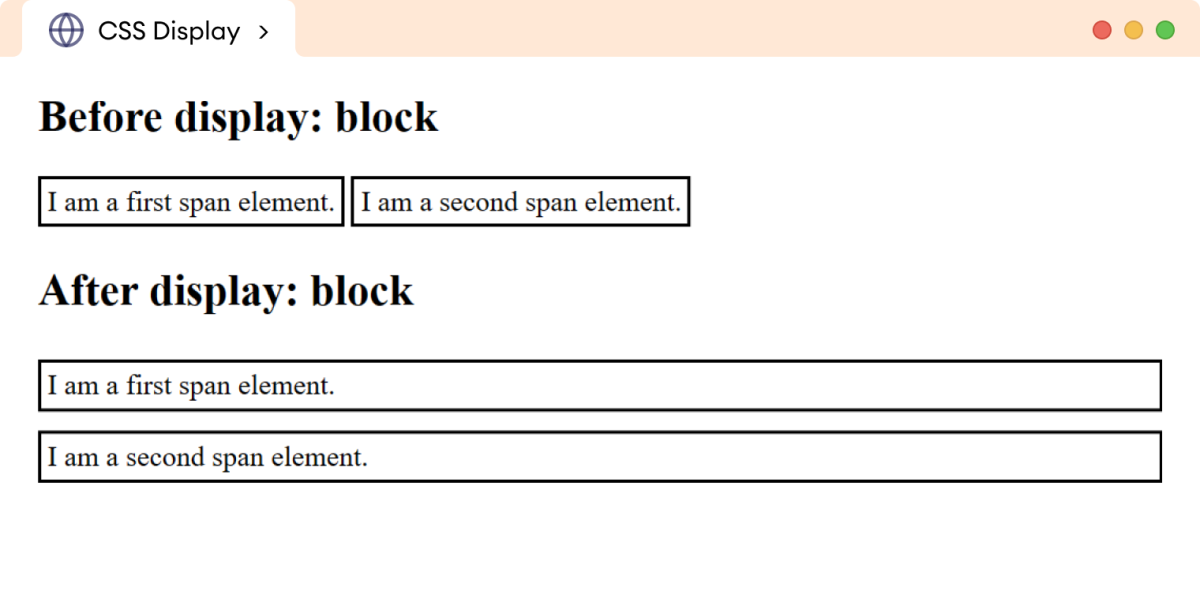
padding: 4px;

}

span.after {

display: block;

}



## Inline -Block:

div {

display: inline-block;

}

Flex:

The flex value of the display property allows for efficient alignment and distribution of the space between the child elements within the parent element.

<body>

<ulclass="parent">

<li>Home</li>

<li>About</li>

<li>Gallery</li>

<li>Blogs</li>

<li>Contact</li>

</ul>

</body>

ul.parent {

display: flex;

background-color: greenyellow;

padding: 0px;

}

li {

background: skyblue;

border: 1px solid black;

padding: 12px;

margin: 8px;

list-style: none;

}

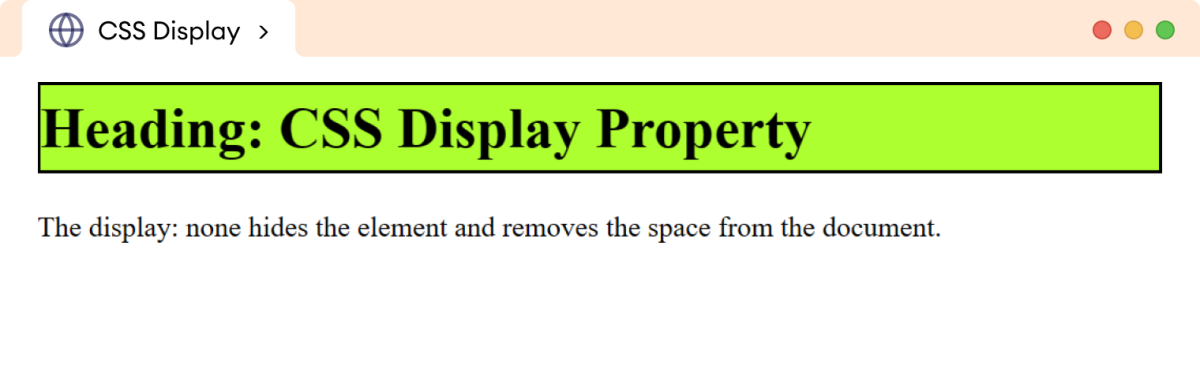
****CSS Display None

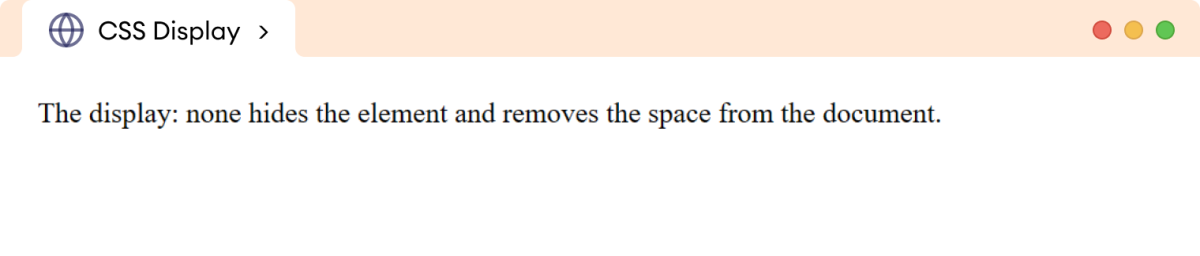
The none value of the display property is used to hide an element and remove it from the normal flow of the document.

h1 {

display: none;

}

****

****

**CSS float Property**

The CSS float property is used to specify how an element floats within a parent element. The element can float to the right, left, or none within the container.

The syntax of the float property is as follows,

* none: element doesn't float (default value)
* left: element floats on the left side of its containing block
* right: element floats on the right side of its containing block
* initial: the value is set to the default value
* inherit: inherits floating property from its parent element

The float property is mainly used for text wrapping around elements. The position property is used to position the element with precise control within the document.

## CSS float none Value

<divclass="parent">

<img

src="https://www.seekpng.com/png/detail/109-1093815\_doraemon-png-png-images-doraemon-png.png"

alt="A flying Doremon image"

/>

<p>

Doraemon is a beloved cartoon character adored by kids all over

the world. He is a robotic cat from the future who travels back

in time to help a young boy named Nobita. With his magical

gadgets and clever solutions, Doraemon always manages to turn

Nobita's troubles into exciting adventures.

</p>

</div>

img {

/\* default value \*/

float: none;

width: 100px;

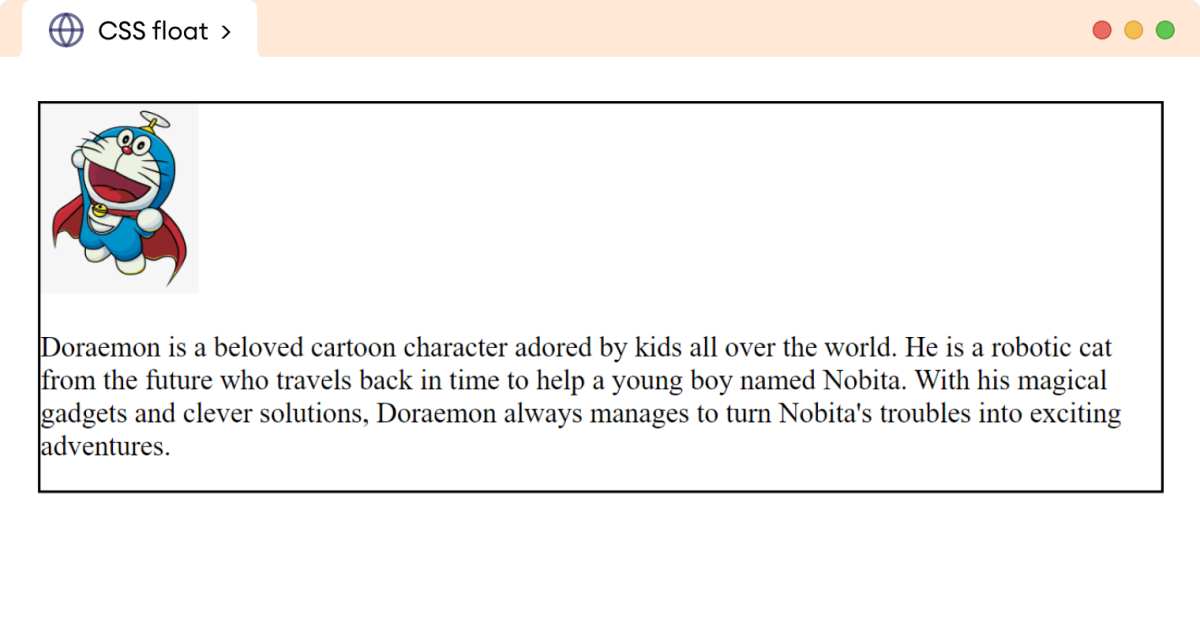
height: 120px;

}

div.parent {

border: 2px solid black;

}

****

## CSS float right Value

img {

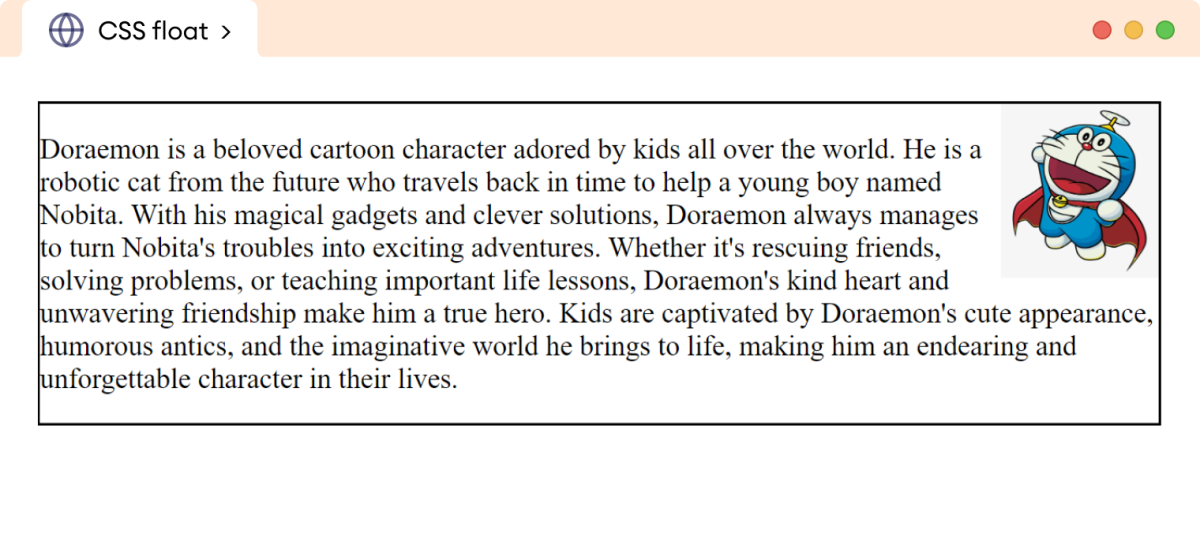
/\* pushes the image to the right of its container/parent \*/

float: right;

width: 100px;

height: 110px;

}

****

## CSS float left Value

img {

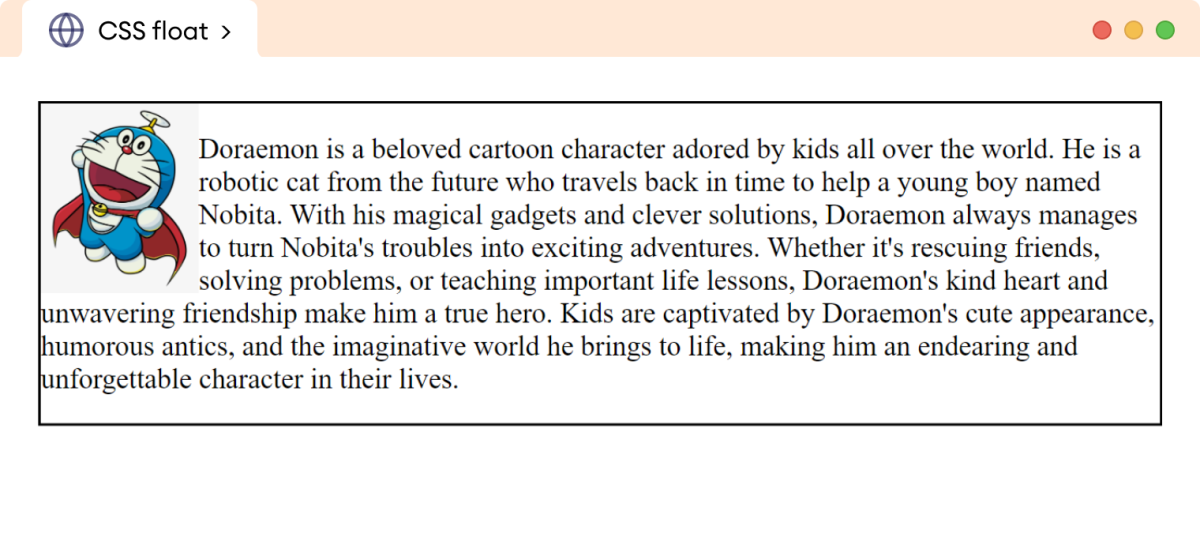
/\* pushes the image to the left of its container/parent \*/

float: left;

width: 100px;

height: 120px;

}

****

**CSS Navigation Bar**

CSS **Navigation Bar**also known as **Nav Bar** helps users navigate through different sections of a website. It is styled using CSS to enhance its appearance and functionality, making it visually appealing and responsive. Typically located at the top or side of a webpage.

<ul>

<li><a href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>



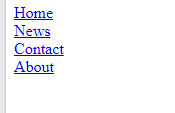
ul {

list-style-type: none;

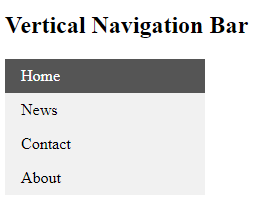
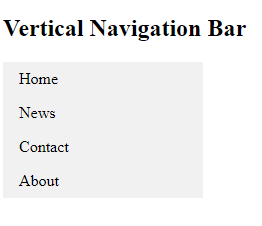
margin: 0;

padding: 0;

}



Vertical Navigation Bar



<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

width: 200px;

background-color: #f1f1f1;

}

li a {

display: block;

color: #000;

padding: 8px 16px;

text-decoration: none;

}

/\* Change the link color on hover \*/

li a:hover {

background-color: #555;

color: white;

}

</style>

## Horizontal Navigation Bar

<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

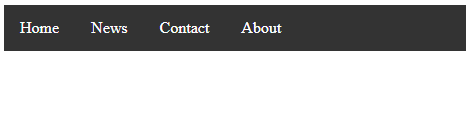
}

li {

display: inline;

}

</style>



<style>

ul {

list-style-type: none;

margin: 0;

padding: 0;

background-color: #333;

}

li {

float: left;

}

li a {

display: block;

color: white;

text-align: center;

padding: 14px 16px;

text-decoration: none;

}

li a:hover {

background-color: #111;

}

</style>

**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.

Instead of using three separate images, we use this single image ("img.gif"):

C:\Users\hp\Desktop\img_navsprites.gif

With CSS, we can show just the part of the image we need.

In the following example the CSS specifies which part of the "img.gif" image to show:

<html>

<head>

<style>

#home {

width: 46px;

height: 44px;

background: url(img.gif) 0 0;

}

#next {

width: 43px;

height: 44px;

background: url(img.gif) 42px 0;

}

</style>

</head>

<body>

<img id="home" src="img\_trans.gif" width="1" height="1">

<img id="next" src="img\_trans.gif" width="1" height="1">

</body>

background: url(img.gif) 0 0; - Defines the background image and its position (left 0px, top 0px)

# CSS Attribute Selectors

Style HTML Elements With Specific Attributes

It is possible to style HTML elements that have specific attributes or attribute values.

<body>

<p class="first">This is a first paragraph.</p>

<p>This paragraph doesn't have a class attribute.</p>

<p class="third">This is a third paragraph.</p>

<p class="fourth">This is a fourth paragraph.</p>

</body>

p[class] {

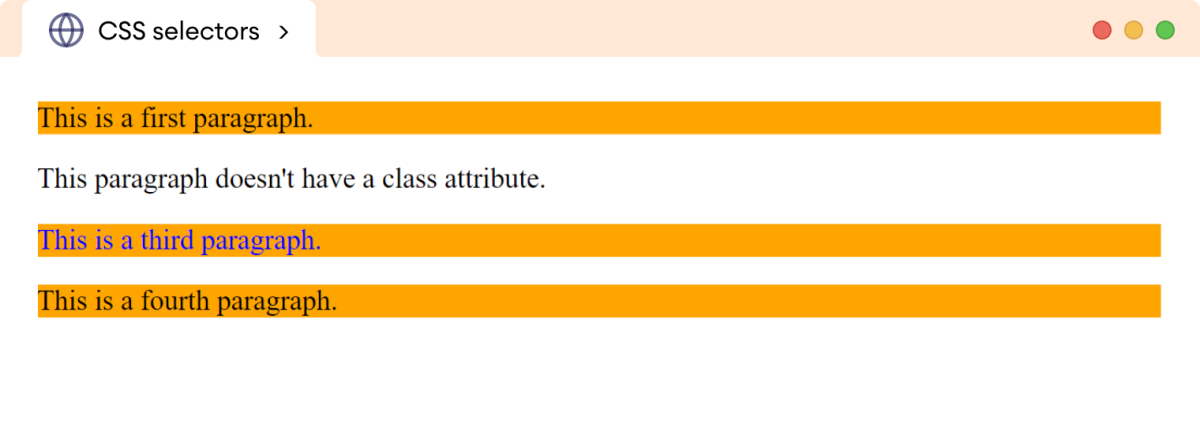
background-color: orange;

}

p[class="third"] {

color: blue;

}



CSS [attribute] Selector

The [attribute] selector is used to select elements with a specified attribute.

<head>

<style>

**a[target] {**

**background-color: yellow;**

**}**

</style>

</head>

<body>

<a href="">Home</a>

**<a href="" target="\_blank">About</a>**

**<a href="" target="\_top">Contact</a>**

</body>

## C:\Users\hp\Desktop\Untitled.png

## CSS [attribute="value"] Selector

The [attribute="value"] selector is used to select elements with a specified attribute and value

<style>

a[target="\_blank"] {

background-color: yellow;

}

</style>



CSS [attribute~="value"] Selector

The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word.

<html>

<head>

<style>

[title~="flower"] {

border: 5px solid yellow;

}

</style>

</head>

<body>

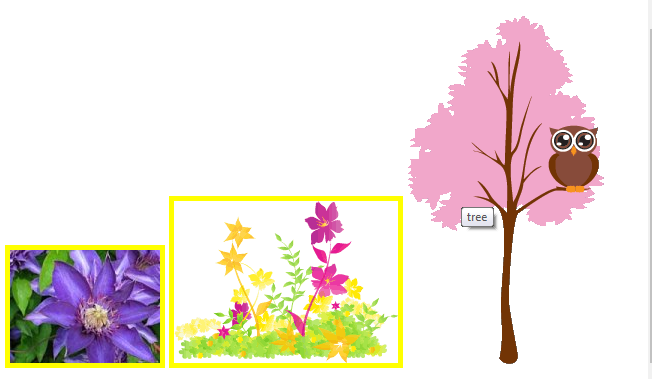
<img src="klematis.jpg" title="klematis flower" width="150" height="113">

<img src="img\_flwr.gif" title="flower" width="224" height="162">

<img src="img\_tree.gif" title="tree" width="200" height="358">

</body>

</html>



CSS [attribute|="value"] Selector

The [attribute|="value"] selector is used to select elements with the specified attribute, whose value can be exactly the specified value, or the specified value followed by a hyphen (-).

<html>

<head>

<style>

[class|="top"] {

background: yellow;

}

</style>

</head>

<body>

<h2>CSS [attribute|="value"] Selector</h2>

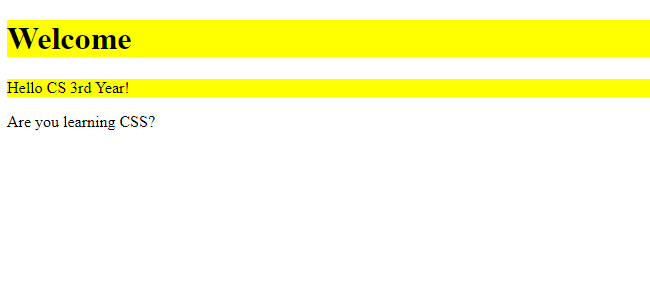
<h1 class="top-header">Welcome</h1>

<p class="top-text">Hello CS 3rd Year!</p>

<p class="topcontent">Are you learning CSS?</p>

</body>

</html>



Styling Forms

The attribute selectors can be useful for styling forms without class or ID:

<head>

<style>

input[type="text"] {

width: 150px;

display: block;

margin-bottom: 10px;

background-color: yellow;

}

input[type="button"] {

width: 120px;

margin-left: 35px;

display: block;

}

</style>

</head>

<body>

<h2>Styling Forms</h2>

<form name="input" action="" method="get">

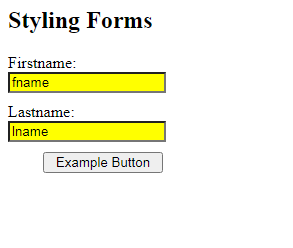
Firstname:<input type="text" name="Name" value="fname" size="20">

Lastname:<input type="text" name="Name" value="lname" size="20">

<input type="button" value="Example Button">

</form>

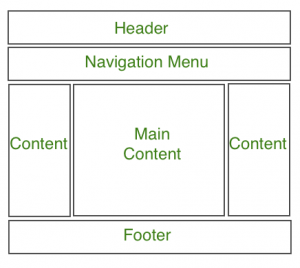
</body>



**Website Layouts**

CSS Website Layout plays a crucial role in defining its visual structure, organization, and responsiveness when designing a website.

A website can be divided into various sections comprising of header, menus, content, and footer based on which there are many different layout designs available for developers. Different layouts can be created by using a div tag and using CSS property to style it.



**1. Header Section**

The header section typically appears at the top of a website or just below the top navigation menu. It often includes the website name or logo.

<**html**>

<**head**>

<**title**>

Website Layouts

</**title**>

<**style**>

.**header** {

**background-color**: **green**;

**padding**: 15px;

**text-align**: **center**;

}

</**style**>

</**head**>

<**body**>

<**div** class="header">

<**h2** style="color:white;">

BSA College of Engineering & Technology, Mathura

</**h2**>

</**div**>

<**br**>

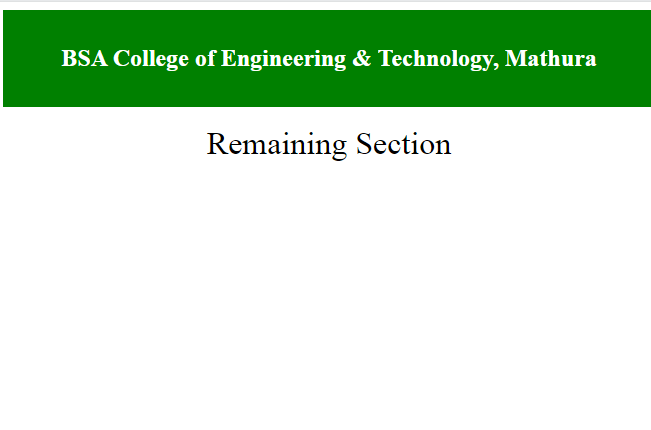
<**center** style="font-size:200%;">

Remaining Section

</**center**>

</**body**>

</**html**>



### **2. Navigation Menu**

A navigation bar/menu provides links for easy website navigation.

.**nav\_menu** {

**overflow**: **hidden**;

**background-color**: #333;

}

.**nav\_menu** **a** {

**float**: **left**;

**color**: **white**;

**text-align**: **center**;

**padding**: 14px 16px;

**text-decoration**: **none**;

}

.**nav\_menu** **a**:hover {

**background-color**: **white**;

**color**: **green**;

}

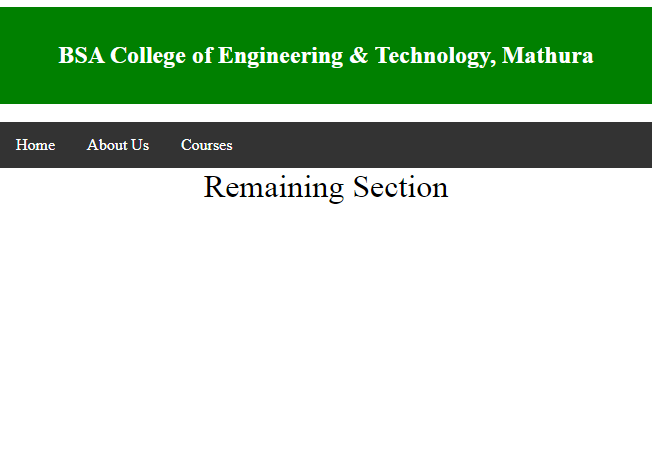
<**div** class="nav\_menu">

<**a** href="#">Home</**a**>

<**a** href="#">About Us</**a**>

<**a** href="#">Courses</**a**>

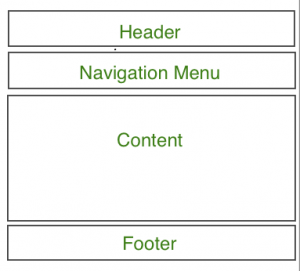
</**div**>



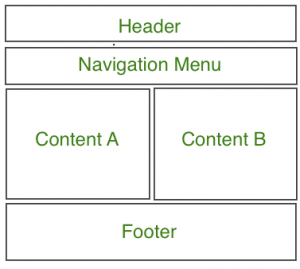
### **3. Content Section**

The content section is the main body of the website. The user can divide the content section in an n-column layout.   
The most common layouts are:

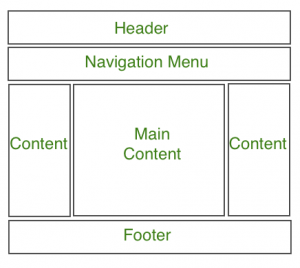
* **1 Column Layout:** It is mostly used for mobile layout.



* **2 Column Layout:** This website layout is mostly used for tablets or laptops.



* **3 Column Layout:** This website layout is mostly used for desktops.



The user can also create a responsive layout where the layout will get changed as per screen size. Consider the below example where if the width of the screen is more than 600px then there will be a 3-column layout and if the width of the screen is between 400px to 600px then there will be a 2-column layout and if the screen size is less than 400px then the 1-column layout will display.

*/\* CSS property for content section \*/*

.**columnA**,

.**columnB**,

.**columnC** {

**float**: **left**;

**width**: 28%;

**padding**: 15px;

**text-align**: **justify**;

}

**h2** {

**color**: **green**;

**text-align**: **center**;

}

*/\* Media query to set website layout*

*according to screen size \*/*

@**media** **screen** **and** (**max-width**:600px) {

.**columnA**,

.**columnB**,

.**columnC** {

**width**: 50%;

}

}

@**media** **screen** **and** (**max-width**:400px) {

.**columnA**,

.**columnB**,

.**columnC** {

**width**: 100%;

}

}

</**style**>

<**div** class="columnA">

<**h2**>Column A</**h2**>

<**p**>NEWS & UPDATES</**p**>

</**div**>

<**div** class="columnB">

<**h2**>Column B</**h2**>

<**p**>UPCOMING EVENTS</**p**>

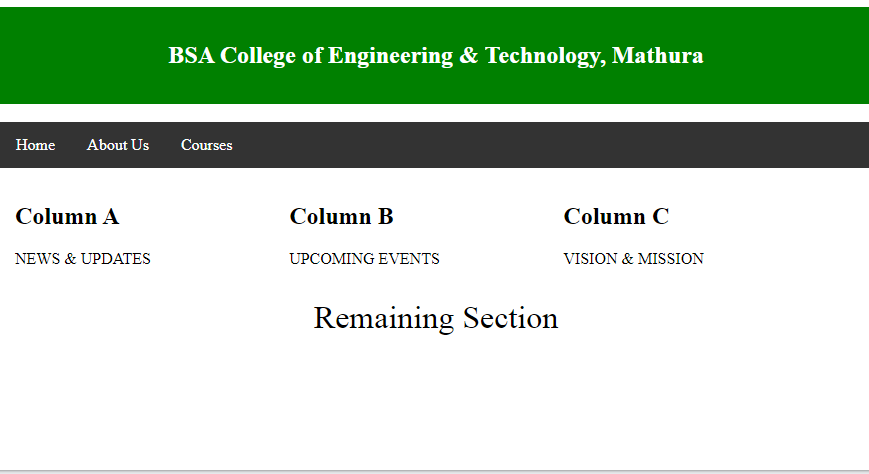
</**div**>

<**div** class="columnC">

<**h2**>Column C</**h2**>

<**p**>VISION & MISSION</**p**>

</**div**>



### **4. Footer Section**

A footer section is placed at the bottom of the webpage and it generally consists of information like contact info, copyrights, About us etc.

<**style**>

*/\* Style for footer section \*/*

.**footer** {

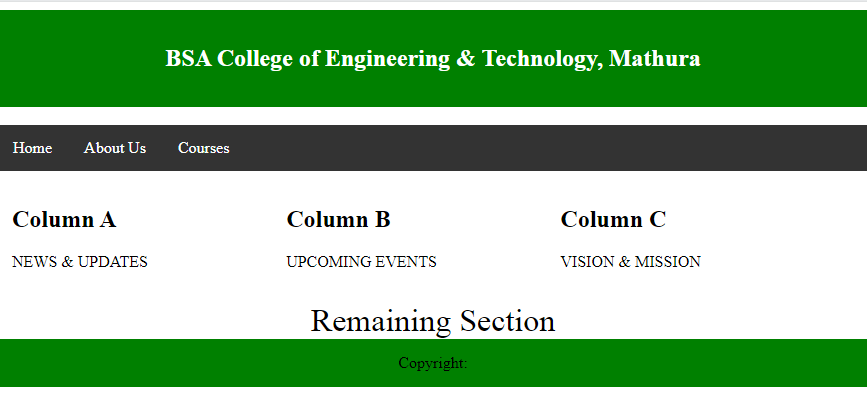
**background-color**: **green**;

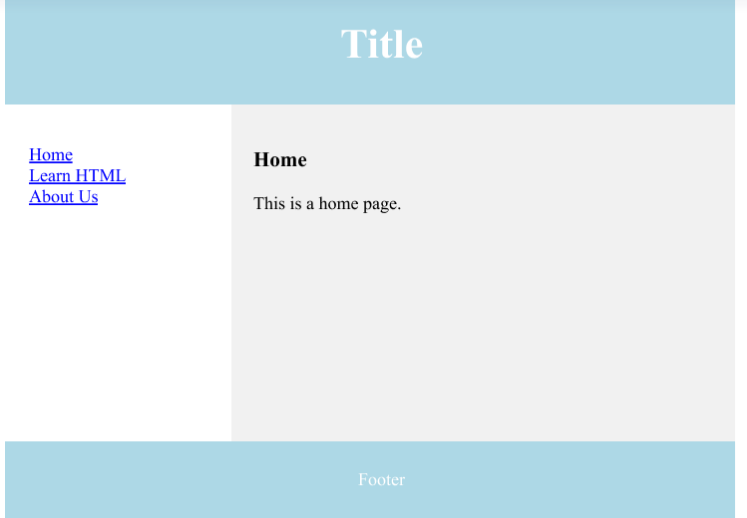
**padding**: 15px;

**text-align**: **center**;

}

</**style**>





**Using Display: flex;**

ul.parent {

display: flex;

background-color: greenyellow;

padding: 0px;

}

li {

background: skyblue;

border: 1px solid black;

padding: 12px;

margin: 8px;

list-style: none;

}



**Using Unordered List**

<html>

<head>

<style>

ul {

overflow:hidden;

list-style-type: none;

margin: 0;

padding: 0;

background-color: blue;

}

li

{

display:inline;

}

li a

{

float:left;

text-decoration: none;

padding: 14px 16px;

color:white;

}

li a:hover {

background-color: #111;

}

</style>

</head>

<body>

<ul>

<li><a class="active" href="#home">Home</a></li>

<li><a href="#news">News</a></li>

<li><a href="#contact">Contact</a></li>

<li><a href="#about">About</a></li>

</ul>

</body>

</html>

