CSS Tutorial

* CSS stands for Cascading Style Sheets.
* It is the language, used to style a document written in a markup language such as HTML or XML.
* CSS saves a lot of work. It can control the layout of multiple web pages all at once.
* External stylesheets are stored in .css files.
* World Wide Web Consortium (W3C) created CSS.

# CSS Syntax:



* 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.

**Types of CSS Selectors:**

1. Simple selectors (select elements based on name, id, class)
2. [Combinator selectors](https://www.w3schools.com/css/css_combinators.asp) (select elements based on a specific relationship between them)
3. [Pseudo-class selectors](https://www.w3schools.com/css/css_pseudo_classes.asp) (select elements based on a certain state)
4. [Pseudo-elements selectors](https://www.w3schools.com/css/css_pseudo_elements.asp) (select and style a part of an element)
5. [Attribute selectors](https://www.w3schools.com/css/css_attribute_selectors.asp) (select elements based on an attribute or attribute value)
6. **Simple Selector:**
7. **The tag Selector:**  styles a html tag based on its name.

Example: p { text-align: center; color: red;}

1. **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. To select an element with a specific id, write a hash (#) character, followed by the id of the element.

Example: CSS -> #id1 { text-align: center; color: red;}

HTML -> <p id=”id1”>test data</p>

1. **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.

For Example: .center { text-align: center; color: red;}

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

For Example: p.center { text-align: center; color: red;}

HTML elements can also refer to more than one class.

Example: <p class="center large">This paragraph refers to two classes.</p>

1. **The Universal Selector**: (\*) selects all HTML elements on the page.

Example: \* { text-align: center; color: blue;}

1. **The Grouping Selector:** selects all the HTML elements with the same style definitions.

Example: h1, h2, p { text-align: center; color: red;}

1. [**Combinator selectors**](https://www.w3schools.com/css/css_combinators.asp)**:**
2. **Descendant combinator (space):** matches all elements that are inside of a specified element.

For Example: div p { background-color: yellow;} style all <p> elements which are nested in <div> element.

1. **Child combinator (>):** matches all elements that are the **direct** children of a specified element. For Example: div > p {background-color: yellow;}
2. **Adjacent sibling combinator (+):** matches an element which is **immediately** following a specific element.

For Example: h1 + h2 { color: blue;} style the <h2> tag which is immediate after a <h1> tag.

1. **Subsequent-sibling combinator (~):** matches all elements which are following a specific element. Example: h1 ~ h2 { color: blue;}
2. **Pseudo-class selectors**

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

For example, it can be used to:

* Style an element when a user moves the mouse over it
* Style visited and unvisited links differently
* Style an element when it gets focus
* Style valid/invalid/required/optional form elements

**Syntax:**

selector:pseudo-class { property: value;}

Some Common types:

1. **:hover** – style an element when user hover over it using their mouse.

Example – button:hover { color:red;}

1. **:active** – style an element when it being activated just as when user click on it.
2. **:visited** – style a link that has already been clicked by the user.

Example – a:visited { color:red;}

**Note:** a:hover MUST come after a:link (to style unvisited link) and a:visited in the CSS definition in order to be effective!

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

1. **:focus** – style a element that currently have focus such as input or enter button.
2. **:first-child** – style a specified element that is the first child of another element or root.
3. **:last-child** – style a specified element that is the last child of another element or root.
4. **:first-of-type** – style first occurence of a specified element type within its parent.
5. **:nth-child(n)** – style a specified element that is the nth child of another element or root.

Example – p:nth-child(3) { color:red;} style 3rd <p> tag in its parent.

p:nth-child(even) { color:red;} style all even positioned (0,2,..) <p> tag in its parent.

**Pseudo-classes and HTML Classes:**

Pseudo-classes can be combined with HTML classes.

Example: a.class\_name:hover { color: #ff0000;} style a link whose class=”class\_name”

**Match the first <i> element in all <p> elements:**

p i:first-child { color: blue;}

**Match all <i> elements in all first child <p> elements:**

p:first-child i {color: blue;}

1. **Pseudo-elements selectors**

A CSS pseudo-element is used to style specific 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 an element
* Style the markers of list items
* Style the viewbox behind a dialog box

**Syntax:**

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

Some Common types:

1. **::first-line** – used to add a special style to the first line of a text.

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

1. **::first-letter** – used to add a special style to the first letter of a text.
2. **::before** – used to insert some content before the content of an element.

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

1. **::after** – used to insert some content after the content of an element.
2. **::marker** – style the markers of list items.

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

1. **Attribute selectors**

Used to style HTML elements that have specific attributes or attribute values.

**Syntax:**

Tag\_name[attribute condition] { property: value;}

Some common types:

1. **[attribute] selector:** used to select elements with a specified attribute.

Example: a[target] { background-color: yellow;} selects all <a> elements with a target attribute:

1. **[attribute="value"] selector**: used to select elements with a specified attribute and value.
2. **[attribute~="word"] selector**: used to select elements with an attribute value containing a specified word.
3. **[attribute|="value"] selector**: used to select elements with the specified attribute, whose value can be exactly the specified value, or the specified value followed by a hyphen (-).
4. **[attribute^="value"] selector**: used to select elements with the specified attribute, whose value starts with the specified value.

Eg: a[href^="https"] Selects all <a> elements with a href attribute value starting with "https".

1. **[attribute$="value"] selector**: used to select elements whose attribute value ends with a specified value.
2. **[attribute\*="value"] selector:** used to select elements whose attribute value contains a specified value.

Example: [class\*="te"] {color: yellow;} selects all elements with a class attribute value that contains "te".

**Three Ways to add CSS:**

1. **External CSS:**

With an external style sheet, you can change the look of an entire website by changing just one file. External styles are defined within the <link> element, inside the <head> section of an HTML page.

Example: test.html -> <link rel="stylesheet" href="mystyle.css">

mystyle.css -> body { background-color: lightblue;} h1 {color: navy;}

1. **Internal CSS:**

An internal style sheet may be used if one single HTML page has a unique style. The internal style is defined inside the <style> element, inside the head section of an HTML page.

Example: <style>  
 body { background-color: linen;}  
 h1 { margin-left: 40px;}  
 </style>

1. **Inline CSS:**

An inline style may be used to apply a unique style for a single element. Inline styles are defined within the "style" attribute of the relevant element.

Example: <h1 style="color:blue;text-align:center;">This is a heading</h1>

**Cascading Order (Priority Order):**

Inline style > External and internal style sheet (calling order) > Browser default

**CSS Comments:**

A CSS comment is placed inside the <style> element, and starts with /\* and ends with \*/.

Example: /\* This is a single-line comment \*/

**CSS Colors:**

In CSS, a Color is specified using predefined color name, or RGB, HEX, HSL, RGBA, HSLA value.

For example: color name "**Tomato**" can be represented as:

* rgb(255, 99, 71)
* #ff6347
* hsl(9, 100%, 64%)

Same as color name "Tomato", but 50% transparent:

* rgba(255, 99, 71, 0.5)
* hsla(9, 100%, 64%, 0.5)

1. **rgb(*red,* *green*, *blue*):**

* Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.
* To display black, set all color parameters to 0, like this: rgb(0, 0, 0).
* To display white, set all color parameters to 255, like this: rgb(255, 255, 255).

1. **rgba(*red,* *green*, *blue, alpha*):**

* RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color.
* Alpha values lies between 0.0 (fully transparent) and 1.0 (not transparent at all).

1. **Hex value:**

* A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color. Each lies between 00 and ff (same as decimal 0-255).
* To display black, set all values to 00, like this: #000000.
* To display white, set all values to ff, like this: #ffffff.
* The 3-digit hex code (#rgb) is a shorthand for some 6-digit hex codes (#rrggbb).

1. **hsl(*hue*, *saturation*, *lightness*):**

* Hue is a degree on the color wheel from 0 to 360. 0 is red, 120 is green, and 240 is blue.
* Saturation is a percentage value. 0% means a shade of gray, and 100% is the full color.
* Lightness is also a percentage. 0% is black, 50% is neither light or dark, 100% is white.

Syntax Example: <p style="color:Tomato;">

**CSS Backgrounds**: used to add background effects for elements.

1. **CSS Background Color:** set the background color for HTML elements.

Example: <p style="background-color:Tomato;">

1. **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.

Example: <style> body {background-image: url("paper.gif"); } </style>

1. **CSS background-repeat:** It is used to repeat an image only horizontally (background-repeat: **repeat-x**;), or only vertically (background-repeat: **repeat-y**;), or **no-repeat**.

Example: body {  
  background-image: url("img\_tree.png");  
  background-repeat: no-repeat;  
  background-position: right top; /\* to specify the position of the background image\*/  
 }

1. **CSS background-attachment:** The background-attachment property specifies whether the background image should **scroll** or be **fixed** (will not scroll with the rest of the page):

Example: background-attachment: fixed;

**Note:** To shorten the code, it is also possible to specify all the background properties in one single property. This is called a **shorthand property.**

Syntax: background: background-color background-image background-repeat background-attachment background-position;

Example: background: #ffffff url("img\_tree.png") no-repeat right top;

**CSS Border properties:** allow you to specify the style, width, and color of an element's border.

1. **CSS Border Style:** The **border-style** property specifies what kind of border to display.

The following values are allowed:

* dotted - Defines a dotted border
* dashed - Defines a dashed border
* solid - Defines a solid border
* double - Defines a double border
* groove - Defines a 3D grooved border. The effect depends on the border-color value
* ridge - Defines a 3D ridged border. The effect depends on the border-color value
* inset - Defines a 3D inset border. The effect depends on the border-color value
* outset - Defines a 3D outset border. The effect depends on the border-color value
* none - Defines no border
* hidden - Defines a hidden border.

1. **CSS Border Width:** The **border-width** property specifies the width of the four borders.

The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: thin, medium, or thick. Default value is medium.

Example: p {  
  border-style: solid;  
  border-width: 5px 20px; /\* 5px top and bottom, 20px on the sides \*/  
 }

1. **CSS Border Color:** The **border-color** property is used to set the color of the four borders. If border-color is not set, it inherits the color of the element.
2. **CSS Border - Shorthand Property:**

The **border** property is a shorthand property for the following individual border properties:

* border-width
* border-style (**required**)
* border-color

Example: p { border: 5px solid red;}

You can also specify all the individual border properties for just one side:

Example: p { border-bottom: 6px solid red;}

1. **CSS Rounded Borders:**

The **border-radius** property is used to add rounded borders to an element.

Example: p {border: 2px solid red;  border-radius: 5px;}

**Note:**

* None of the OTHER CSS border properties will have ANY effect unless the border-style property is set!
* The border-style, border-width and border-color properties can have from one to four values (for the top border, right border, bottom border, and the left border).

**CSS Margins:**

Margins are used to create space around elements, outside of any defined borders.

CSS has properties for specifying the margin for each side of an element:

* margin-top
* margin-right
* margin-bottom
* margin-left

All the margin properties can have the following values:

* auto - to horizontally center the element within its container.
* *length* - specifies a margin in px, pt, cm, etc.
* *%* - specifies a margin in % of the width of the containing element
* inherit - specifies that the margin should be inherited from the parent element

**Tip:** Negative values are allowed.

To shorten the code, it is possible to specify all the margin properties in one property using ***margin***.

margin property has different number of values:

* margin: 25px 50px 75px 100px; = top, right, bottom, left;
* margin: 25px 50px 75px; = top, right and left, bottom;
* margin: 25px 50px; = Top and bottom margins are 25px, right and left margins are 50px
* margin: 25px; = all four margins are 25px;

Example: p {margin: 25px 50px;}

**CSS Padding:**

Padding is used to create space around an element's content, inside of any defined borders.

Similar to margin, we have properties for setting the padding for each side of an element (top, right, bottom, and left).

Example: div { padding: 25px 50px 75px 100px;}

**CSS height and width:**

The **height** and **width** properties are used to set the height and width of an element.

The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

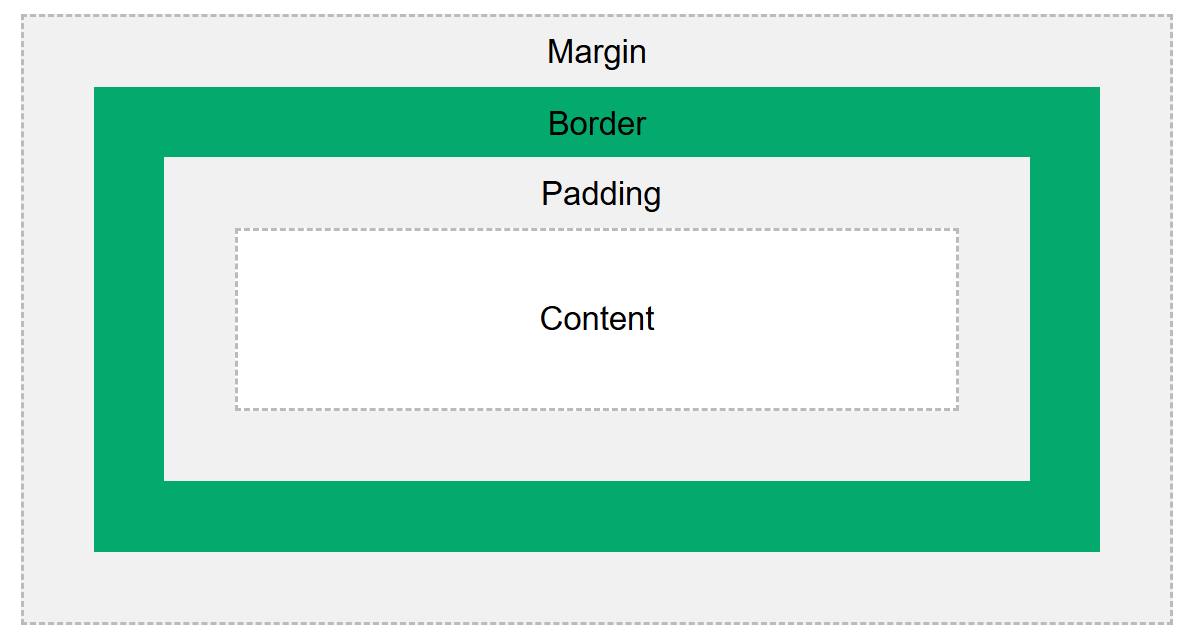
* max-height = Sets the maximum height of an element
* max-width = Sets the maximum width of an element
* min-height = Sets the minimum height of an element
* min-width = Sets the minimum width of an element

Example: div {height: 200px; width: 50%; }

**CSS Box Model:**

It is essentially a box that wraps around every HTML element. It consists of: content, padding, borders and margins.

**Note:** When you set the width and height properties of an element with CSS, you just set the width and height of the **content area**. To calculate the total width and height of an element, you must also include the padding and borders.



**CSS Box Sizing:**

Defines how the width and height of an element are calculated: should they include padding and borders, or not.

If you set box-sizing: border-box; on an element, padding and border are included in the width and height:

Example: div {  
  width: 300px;  
   padding: 25px;  
   box-sizing: border-box; }

**CSS Outline:**

An outline is a line drawn outside the element's border.

CSS has the following outline properties (similar to border-properties):

* outline-style
* outline-color
* outline-width
* outline-offset: adds space between an outline and the edge/border of an element.
* outline: outline-width outline-style outline-color;

**CSS Text**

1. **Text Color:** The color property is used to set the color of the text.

For Example: h1 { color: green; }

1. **Text Alignment:** The **text-align** property is used to set the **horizontal alignment** of a text. A text can be left or right aligned, centered, or justified. Left alignment is default if text direction is left-to-right and vice versa.

When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers).

Eg. div { text-align: justify; }

The **text-align-last** property specifies how to align the last line of a text.

Eg. p {  text-align-last: right;} /\* align last line of paragraph to right \*/

The **vertical-align** property sets the **vertical alignment** of an element.

Eg. img.b { vertical-align: text-top;} /\* align image with class=’b’ to the top of text.

1. **Text Decoration:**

* The **text-decoration-line** property is used to add a decoration line to text. You can combine more than one value, like overline and underline to display lines both over and under a text.

Eg. p { text-decoration-line: overline underline; }

* The **text-decoration-color** property is used to set the color of the decoration line.
* The **text-decoration-style** property is used to set the style of the decoration line.
* The **text-decoration-thickness** property is used to set the thickness of the decoration line.

Eg. p { text-decoration-line: underline;

text-decoration-color: red;

text-decoration-style: wavy;

text-decoration-thickness: 5px; }

* The **text-decoration property** is a shorthand property for: text-decoration-line (required), text-decoration-color (optional), text-decoration-style (optional), text-decoration-thickness (optional).

Eg. p {  text-decoration: underline red double 5px;}

1. **Text Transformation:** The text-transform property is used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word.

Eg. p {  text-transform: uppercase;}

1. **Text Spacing:**

* The **text-indent** property is used to specify the indentation of the first line of a text:
* The **letter-spacing** property is used to specify the space between the characters in a text.
* The **line-height** property is used to specify the space between lines. The default line height in most browsers is about 110% to 120%.
* The **word-spacing** property is used to specify the space between the words in a text.
* The **white-space** property specifies how white-space inside an element is handled.

Eg. p{

text-indent: 5px;

letter-spacing: 2px;

line-height: 0.7;

word-spacing: 10px;

white-space: nowrap; }

1. **Text Shadow:** The **text-shadow** property adds shadow to text.

text-shadow: horizontal vertical blur color;

Eg. h1 {  text-shadow: 2px 2px 5px red;}

**CSS Fonts**

1. **Font Families**

In CSS there are five generic font families:

1. **Serif** fonts have a small stroke at the edges of each letter. They create a sense of formality and elegance.
2. **Sans-serif** fonts have clean lines (no small strokes attached). They create a modern and minimalistic look.
3. **Monospace** fonts - here all the letters have the same fixed width. They create a mechanical look.
4. **Cursive** fonts imitate human handwriting.
5. **Fantasy** fonts are decorative/playful fonts.

In CSS, we use the **font-family property** to specify the font of a text. If the font name is more than one word, it must be in quotation marks.

Example:

p {  font-family: "Times New Roman", Times, serif;}

In above example, If the first font "Times New Roman" does not work, the browser will try the next one, and the next one, and so on. So, always end the list with a generic font family name.

1. **Font Style:** The **font-style property** is mostly used to specify italic text.

Example: p { font-style: italic; }

1. Font Weight: The font-weight property specifies the weight of a font:

Example: p { font-weight: bold;}

1. **Font Size:** The **font-size** property sets the size of the text. The font-size value can be an absolute, or relative size.

* Absolute size sets the text to a specified size.

Example: h1 {  font-size: 40px;}

* Relative size sets the size relative to surrounding elements.

Example: h1 {  font-size: 2.5em; /\* 40px/16=2.5em \*/ }

* If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).
* **Responsive Font Size**: The text size can be set with a **vw** unit, which means the "viewport width". That way the text size will follow the size of the browser window.

Example: <h1 style="**font-size: 10vw**">Hello World</h1> /\*1vw = 1% of viewport width\*/

1. **Google Fonts:** If you do not want to use any of the standard fonts in HTML, you can use Google Fonts. Just add a special style sheet link in the <head> section and then refer to the font in the CSS.

Example:

<head>  
 **<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">**  
 <style>  
 body {  
  font-family: "Sofia", sans-serif;  
 }  
 </style>  
</head>

**CSS Icons:**

* Icons can easily be added to your HTML page, by using an icon library such as Font Awesome, Bootstrap Icons, and Google Icons.
* Add the name of the specified icon class to any inline HTML element (like <i> or <span>).
* All the icons in the icon libraries are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

1. **Font Awesome Icons:**

To use the Font Awesome icons, go to [fontawesome.com](https://fontawesome.com/), sign in, and get a code to add in the <head> section of your HTML page:

<script src="https://kit.fontawesome.com/*yourcode*.js" crossorigin="anonymous"></script>

Example:

<i class="fas fa-heart"></i>

**Note:** No downloading or installation is required.

**CSS Lists:**

The CSS list properties allow you to:

* Set different list item markers for ordered lists (OL).
* Set different list item markers for unordered lists (UL).
* Set an image as the list item marker
* Add background colors to lists and list items

1. The **list-style-type** property specifies the type of list item marker like **circle, square**, **upper-roman, lower-alpha**, **none** etc..

Example:

ul {  list-style-type: disc;}

ol { list-style-type: decimal-leading-zero;}

1. The **list-style-image** property specifies an image as the list item marker.

Example

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

1. The **list-style-position** property specifies the position of the list-item markers (bullet points).

Example:

ul {  list-style-position: inside;}

1. The **list-style** property is a shorthand property. The order of the property values are:

* **list-style-type** (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
* **list-style-position** (specifies whether the list-item markers should appear inside or outside the content flow)
* **list-style-image**

**CSS Layout - The display Property:**

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

Some common display Property Values:

|  |  |
| --- | --- |
| **Value** | **Description** |
| inline | Displays an element as an inline element |
| block | Displays an element as a block element |
| contents | Makes the container disappear, making the child elements children of the element the next level up in the DOM |
| flex | Displays an element as a block-level flex container |
| grid | Displays an element as a block-level grid container |
| inline-block | Displays an element as an inline-level block container. The element itself is formatted as an inline element, but you can apply height and width values. |

**Display: none;** is commonly used with JavaScript to hide and show elements without deleting and recreating them.

**visibility:hidden**; also hides an element.However, the element will still take up the same space as before. The element will be hidden, but still affect the layout

**CSS Layout - The position Property:**

The **position property** specifies the type of positioning method used for an element (**static, relative, fixed, absolute or sticky**). Elements are then positioned using the **top, bottom, left, and right** properties.

1. **position: static;**

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

1. **position: relative;**

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

1. **position: fixed;**

* An element with position: fixed; is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element.
* A fixed element does not leave a gap in the page where it would normally have been located.

1. **position: absolute;**

* An element with position: absolute; is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed).
* However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

1. **position: sticky;**

* An element with position: sticky; is positioned based on the user's scroll position.
* A sticky element toggles between relative and fixed, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).
* You must specify at least one of top, right, bottom or left for sticky positioning to work.

Example:

div.relative {

position: relative;

width: 400px;

height: 200px;

border: 3px solid #73AD21;

}

div.absolute {

position: absolute;

top: 80px;

right: 0;

border: 3px solid #73AD21;

}

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

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

</div>

**CSS Layout - The z-index Property**

* When elements are positioned, they can overlap other elements.
* 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 can have a positive or negative stack order.
* z-index only works on positioned elements and flex items (elements that are direct children of display: flex elements).
* An element with greater z-index is always above an element with a lower z-index.

Example: img {

position: absolute;

left: 0px;

top: 0px;

z-index: -1;

}

**CSS Layout - Overflow**

The **overflow** property specifies whether to clip the content or to add scrollbars when the content of an element is too big to fit in the specified area.

The overflow property has the following values:

* visible - Default. The overflow is not clipped. The content renders outside the element's box
* hidden - The overflow is clipped, and the rest of the content will be invisible
* scroll - The overflow is clipped, and a scrollbar is added to see the rest of the content
* auto - Similar to scroll, but it adds scrollbars only when necessary

**Note:** The overflow property only works for block elements with a specified height.

The **overflow-x** and **overflow-y** properties specifies whether to change the overflow of content just horizontally or vertically (or both).

The **overflow-wrap** property specifies whether or not the browser can break lines with long words, if they overflow the container.

**CSS Layout - float and clear**

The **float** property is used for positioning and formatting content e.g. let an image float left to the text in a container.

The float property can have one of the following values:

* left - The element floats to the left of its container
* right - The element floats to the right of its container
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

The **clear** property specifies what should happen with the element that is next to a floating element.

The clear property can have one of the following values:

* none - The element is not pushed below left or right floated elements. This is default
* left - The element is pushed below left floated elements
* right - The element is pushed below right floated elements
* both - The element is pushed below both left and right floated elements
* inherit - The element inherits the clear value from its parent