

Styling with CSS

COLOR

The `color` property is used to set the decoration and **foreground color of an element's text context**.

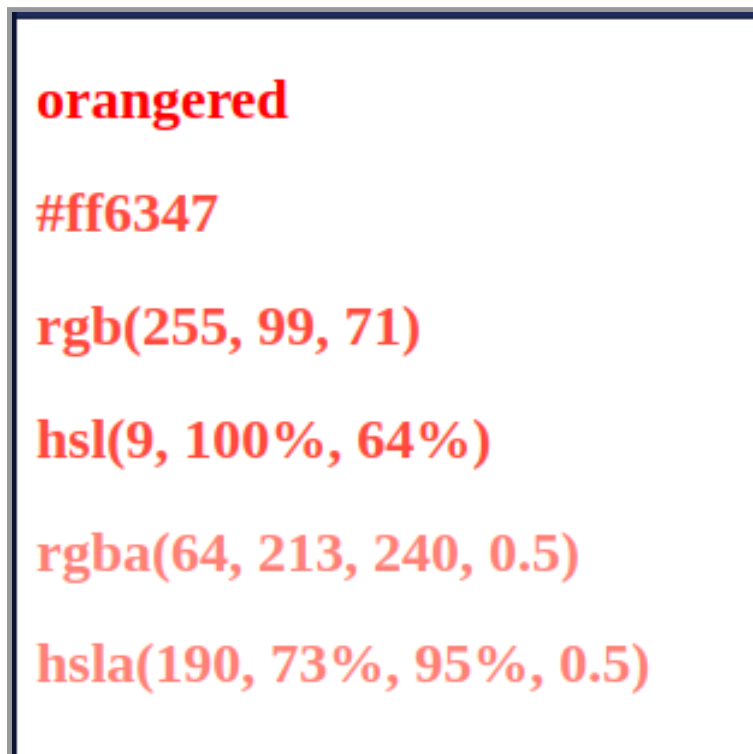
The `color` property can be mentioned in 6 different ways. Each of the given way provides some difference from the other.

Although the `color` property can also be used for the backgrounds and borders, we will discuss them later.

Eg. the following code:

```
<h2 style="color:orangered;">orangered</h2>
<h2 style="color:#ff6347;">#ff6347</h2>
<h2 style="color:rgb(255, 99, 71);">rgb(255, 99, 71)</h2>
<h2 style="color:hsl(9, 100%, 64%;">hsl(9, 100%, 64%)</h2>
<h2 style="color:rgba(255, 99, 71, 0.7);">rgba(255, 99, 71, 0.7)</h2>
<h2 style="color:hsla(9, 100%, 64%, 0.7);">hsla(9, 100%, 64%, 0.7)</h2>
```

shows the following output, which makes the difference quite easy to understand:



- By name

140 different colors named in CSS are supported by all the modern browsers. Unlike HTML, unknown keywords are totally ignored by CSS.

The keywords of colour are all simple, solid colours, without transparency. Eg., *orangered, green, blue, lightgrey, etc.*

- Using RGB

RGB (Red, Green, and Blue) is an abbreviation for **R**ed, **G**reen, and **B**lue. It's a colour model in which a colour is created by **combining red, green, and blue**. The intensity of each color has values ranging from 0 to 255. This gives us a very large number of colors in the dataset.

Eg., the RGB value for **black** is: `rgb(0, 0, 0)` and for **white** is: `rgb(255, 255, 255)`.

- By hex code

The colors can also be given by **6 digits hexadecimal code**. The hex codes are made with the help of the 3 *colors(Red, Green, and Blue)*. The first 2 digits represent red, the next 2 are for green and the last 2 are blue. So, the syntax for hex code can be given as `#RRGGBB`.

Each hexadecimal code value between **00 - FF** is similar to **0 - 255**.

Eg., `#000000` represents **black** and `#FFFFFF` is **white**.

- Using HSL

The HSL (Hue, Saturation, and Lightness) components can also be used to specify the colour.

- **Hue** is a number that ranges from **0** to **360** on the colour wheel. **0** represents **red**, **120** represents **green**, and **240** represents **blue**.
- The amount of saturation in a colour is represented by **saturation**. It's a percentage value; 0% represents a shade of **grey**, and 100% represents the **entire colour spectrum**.
- The amount of **light** in a color is represented by its **lightness**. It's also a percentage, with 0% being **black**, 50% being **neither light nor dark**, and 100% being **white**.

- Using rgba

RGBA (**R**ed, **G**reen, **B**lue, **A**lpha) is a colour space that combines **RGB** and **alpha transparency**. The opacity of the RGB-defined colour is determined by this alpha value. The alpha parameter is a number that ranges from 0.0 (transparent) to 1.0

(opacity) (opaque). For example, `rgba(255, 0, 0, 0.6)` is a red colour, and with 0.6 opacity, it will appear as:



- Using **hsla**

Similar to RGBA, **HSLA** (**H**ue, **S**aturation, **L**ightness, **A**lpha) is a **variant** of HSL that includes **alpha transparency**. The property and alpha value are the same as in RGBA. For example, `hsla(0, 100%, 50%, 0.6)` is also a red colour with 0.6 opacity and the same colour:



CSS UNITS

In CSS, a unit can be represented with **a value followed by the unit symbol**.

For eg: 10px, 5mm, 2%, etc.

Some units are shared between properties, while some are restricted to certain properties.

The unit can be omitted if the value is 0. Whitespace should be omitted between the number and the unit.

Negative lengths are allowed for some CSS properties. For instance, consider the margin property. Absolute and relative length units are the two types of length units.

- **Absolute Units**

The absolute units are a fixed size/length of the element. Because screen sizes vary so much, absolute length units are not recommended to be used on screens.

Absolute units are made up of the following:

- **mm** - millimeters
- **px** - pixels
- **pt** - points
- **pc** - picas
- **cm** - centimeters

- **Relative Units**

Relative length units describe how long something is in comparison to another length property. The following are some examples of relative units:

- **em** - In relation to the parent element's font size (3em means 3 times the size of the current font)
- **rem** - In relation to the root element's font-size

- **vw** - The width of the browser window is about 1% of the total browser width.
- **vh** - Approximately 1% of the browser window's height.
- **%** - With respect to the parent element.

BORDER

The element's border is controlled by the **border** property. The **style**, **width**, and **color** of an element's border can all be specified with CSS borders. Border-width, border-style, and border-color are written together as the **border** which is a **shorthand** for all the properties mentioned above.

Eg., applying border property to a div like this:

```
border: 4px solid red;
```

will show like:

This is border in CSS

We will look into the different properties of the border.

- **Border Width:** The width of the four borders is determined by the **border-width** property. The width can be given by using one of the three predefined values: *thin*, *medium*, or *thick* or as an absolute or relative size Eg., `border-width: 3px 20px 5px 10px;`, will have top border of 3px, right border of 20px, bottom border to 5px and left border to 10px. 3.2.
- **Border Style:** This property determines the type of border that will be displayed. Dotted, solid, dashed, ridge, double, groove, none, inset, outset, hidden are the border-style values.
Eg., `border-style: dotted dashed solid double;` will have dotted top border, dashed right border, solid bottom border, and double lined left border.
NOTE: *It is necessary to add border-style property else no other border properties will work.*
- **Border Color:** The color of the four borders is specified by the **border-color** property. The property's value is identical to that of the **color property**. However, different colors can now be assigned to different border sides. If border-color isn't specified, the element's color is used.

Eg., `border-color: red blue;` The top and bottom borders will be red, while the left and right borders will be blue.

- **Border Individual Sides:** We can provide width, style, and color to each border separately using the border property, but we must still give some value to each side of the border.

CSS border also has the ability to assign a border value to each of the border sides separately. The sides' border properties are:

- **border-top**
- **border-right**
- **border-bottom**
- **border-left**

This is further broken down to give each border side its own style, width, and color. **border-top-style, border-right-width, border-left-color,** and so on are some of them.

- **Border Radius:** This property is used to give an element rounded borders. This property can have an absolute (e.g. in px) or relative (e.g. in percent) value.

Eg., If we add `border-radius: 10px;` the first border example will show as follows:

This is border in CSS

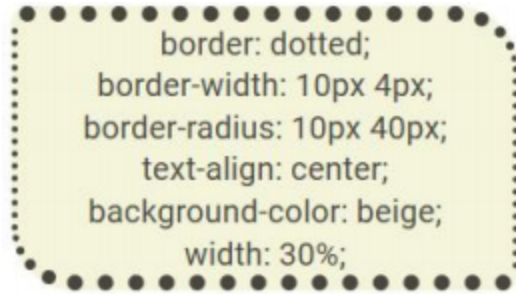
The border-radius can also be in the **elliptical form**. That is why, you need to specify horizontal and vertical radius differently. This is done with the help of a slash ("/") between horizontal and vertical radius.

Here is an example:

```
div {
  border-radius: 40px/20px;
  background: teal;
  width: 20%;
  height: 20%;
  padding: 40px;
}
```



Here is another interesting example:



TEXT AND FONT STYLING

To change the look and style of text in the HTML document, various properties are specified. These styles will apply only to the text content of any element. Let us have a look on some of the most used text and font styling properties.

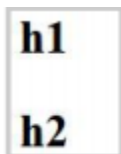
Font properties such as font-family, boldness, size, and style define how a font looks. **Text properties** are used to define the presentation and layout of the text on the HTML page.

- **font-size:** This **sets the text size**. The font-size value can be either absolute or relative, i.e., values in px, percent, em, and so on.

Eg.,

```
h1 {
  font-size: 30px;
}
h2 {
  font-size: 1.875em; }
```

will show




- **font-family:** The font family of a text helps us to set the **font-family** property. As a "fallback" system, the font-family property holds several font names. If the first font isn't supported, the browser moves on to the next font, and so on.

Begin with your desired font and end with a generic family to let your browser select a similar font from the generic family, unless there are other fonts.

Eg.,

```
h1 {
  font-family: sans-serif, monospace, serif;
}
```

will show



NOTE: If a font family's name contains more than one word, it must be enclosed in quotation marks, such as "Times New Roman."

- **font-weight:** The **font-weight** property is used to define the **weight/thickness** of the given font. The weight lies between light to bold. Bold, bolder, inherit, initial, lighter, normal, and unset are all possible values. Alternatively, we can define the font weight using numeric values ranging from 100 to 900.

Eg.,

```
h1 {
  font-weight: lighter;
}
```

will show



- **font-style:** The **font-style** property is used to define the **style for a text**. The general values for this property are: **normal, oblique, italic, initial, inherit**.
- **color:** We have already discussed applying **color** to text using the color property. The color can be defined either by name, hex code, rgb, rgba, hsl, or hsla.
- **text-align:** The **text-align** property is used to **define the horizontal alignment of a text**. A text can be aligned left, right, centered, or justified.

Eg.,

```
p#center {
  text-align: center;
}
p#left {
  text-align: left;
}
p#right {
```

```
text-align: right;
}
```

will show:

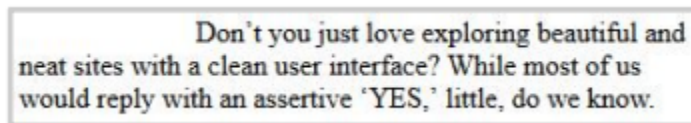


- **text-indent:** The **text-indent** property controls the **indentation** of a **text block's first line**. **Negative numbers** are **permitted**. If the value is negative, the first line will be indented to the left.

Eg.,

```
<p style="text-indent: 100px;">Don't you just love exploring beautiful and neat sites with a clean user interface? While most of us would reply with an assertive 'YES,' little, do we know.</p>
```

will show:



- **text-transform:** The **text-transform** property defines the **case of the letters in a text**. It can be used to turn text to:
 - **uppercase** - this will turn every character to uppercase
 - **lowercase** - this will turn every character to lowercase
 - **capitalize** - this will turn each word's first letter to uppercase, while the rest is converted to lowercase.
 - **none** - It's the default setting. The text is rendered exactly as it is.
- **text-decoration:** The **text-decoration** property is used to **add and remove text decorations**. *To remove underlines from links*, the value **text-decoration: none;** is frequently used. The text-decoration is used to decorate the text. It has 4 values:
 - **underline** - puts a line under the text
 - **overline** - puts a line above the text
 - **line-through** - puts a line through the text
 - **none** - removes any of the above decorations

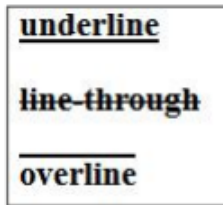
Eg.,

```
h3#underline {
  text-decoration: underline;
}
```



```
h3#line-through {
  text-decoration: line-through;
}
h3#overline {
  text-decoration: overline;
}
```

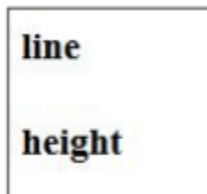
will make the text show like:



- **line-height:** The **line-height** property controls the **height of an element's lines**. It has **no effect** on the **font's size**.

The font-size value can be an absolute or relative size, i.e., values can be applied in px, %, em, etc. The value is multiplied by the font-size of the element if no unit is specified.

Eg., will show the lines like:

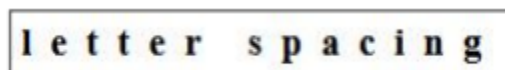


- **letter-spacing:** If you want to **modify the space between the letters**, you can use the **letter-spacing** property. The space between the letters can be **increased** or **decreased** as per convenience.

Eg.,

```
h3 { line-height:1.5; }
```

will show the text like:



- **word-spacing:** The **word-spacing** property can be used to change the amount of space between words.. You can use this to **increase** or **decrease** the space between the words. The value of word-spacing can be absolute or relative.

Eg.,

```
h3 {word-spacing: 10px; }
```

will show the text like:

word spacing used

- **text-shadow:** The **text-shadow** property **adds a shadow to text**. The **position of the horizontal shadow**, the **position of the vertical shadow**, and the **colour of the shadow** are all contained within this value.

Eg.,

```
h2 { text-shadow: 2px 1px red; }
```

will show the text like:

text shadow

BACKGROUND

You can also edit and modify the background of an element. The background includes an **element's dimensions, the padding and border** but **excludes the margin**. Backgrounds in CSS can be **colors** or they can be **images**.

CSS Background properties:

- **background-size**
- **background-attachment**
- **background-image**
- **background-repeat**
- **background-position**

Eg., the below CSS code when applied to a web page

```
body {
  background-image:
url("https://blog.codingninjas.in/wpcontent/uploads/2017/01/cropped-Final_logo_switchtocode-01.png"), lineargradient(#a3f7ff, #fff58e);
```

```
background-repeat: repeat-x;
background-position: center center;
background-attachment: fixed;
}
```

will show the web page like this:



NOTE: You can see about other background properties from <https://developer.mozilla.org/en-US/docs/Web/CSS/background>

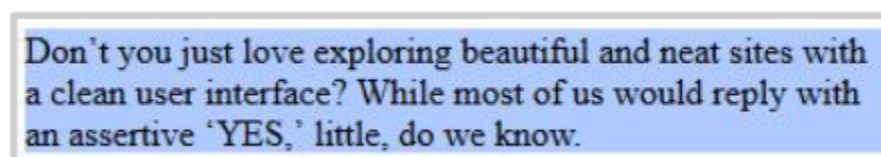
- background-color

The **background-color** property specifies the **color of an element's background**. Its value is the **same as that of the color property**.

Eg.,

```
<p style="background-color: #afcbff;"> Don't you just love exploring beautiful and neat sites with a clean user interface? While most of us would reply with an assertive 'YES,' little, do we know. </p>
```

will show like:



- background-image

To **specify an image to use as the background of an element** the **background - image** property is used .

This property can be used to give an element one or more background images.

By default, a background-image is placed at the top-left corner of an element, and it is repeated so that it covers the entire element in both vertical and horizontal directions.

The values it can take are:

- **url('URL')** - This specifies the image's URL. You can specify multiple images by using a comma to separate the URLs.
- **none** - This is the normal setting. There will be no background image.
- **linear-gradient()** - As the background image, this creates a linear gradient. A minimum of two colors will have to be mentioned (default direction is top to bottom).
- **radial-gradient()** - As the background image, this creates a radial gradient. A minimum of two colors must be mentioned (the default is from the centre to the edges).
- **repeating-linear-gradient()** - repeats a linear gradient
- **repeating-radial-gradient()** - repeats a radial gradient

- background-repeat

You can use the **background-repeat** property to control **how** and **if** a background image is replicated.

By default, a background image repeats both **vertically** and **horizontally**, but **background-repeat** allows you to control how the image repeats.

The values this property can take are

- **repeat** - This is the standard setting. Both vertically and horizontally, the background image is repeated. If the last image does not fit, it will be cropped.
- **repeat-x** - Only the horizontal portion of the image is repeated.
- **repeat-y** - Only the vertical portion of the image is repeated.
- **no-repeat** - the image will only be shown once

- **space** - There is no clipping on the background image. With the first and last images pinned to the sides of the element, the remaining space is evenly distributed between the images.
- **round** - this makes the image to be repeated and shrink or stretch to fill the space

- **background-position**

A background image's initial position is specified using the **background-position** property.

The **background-position** property can be used to **change the position of a background image**, which is by **default in the top-left corner** of an element.

The values this property can take are(**X represents horizontal position and Y represents vertical position**):

- **X Y** - they both can each take value from one of the following - ***left, right, top, bottom, center***. If only one value is specified, the default value is "centre."
- **Xpos Ypos** - specifies the horizontal and vertical position relative to the viewport. Any of the CSS units can be used as units. If only one value is specified, the other value will be set to 50%.

- **background-size**

The **background-size** property is used to **define the image size for the background**.

The values it can take are

- **auto** - This is the default setting. The image is shown in its original dimensions
- **length** - sets the background image's width and height. The width is determined by the first value, while the height is determined by the second value.
- **percentage** - sets the background image's width and height in percent. The width is determined by the first value, while the height is determined by the second. The second value is set to "auto" if only one is provided.
- **cover** - resizes the background image to fill the container's horizontal width
- **contain** - ensures that the background image is fully visible by resizing it

- **background-attachment**

This property determines whether a background image is fixed or scrolls with the rest of the page.

The values it can take are:

- **scroll** - this is the default value. The background image will follow the page as it scrolls.
- **fixed** - The background image will not follow the page as it scrolls.
- **local** - The background image will scroll along with the content of the element.

MARGIN

The CSS margin properties are used to create **space between the borders and the other surrounding elements**.

You can also provide **negative margins** as well.

There are two other values that are used for the margin:

auto - the margin is applied by the browser itself only to horizontal margins

none - to remove any margins from the element or makes the value of margin equal to zero

Just like the border property, you can provide margins separately to the sides. The margin property for the sides are:

- margin-bottom
- Margin-left
- margin-top
- margin-right

Eg. for the below HTML code:

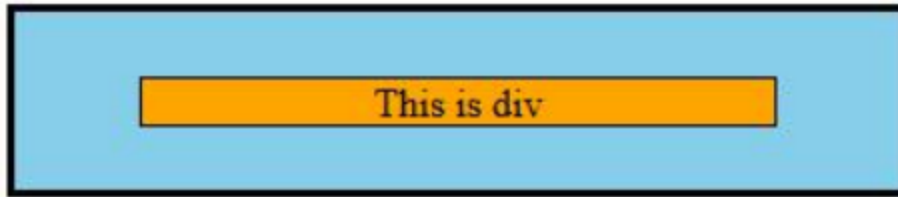
```
<div class="blue-box"><div class="orange-box">This is div</div></div>
```

and applying the following CSS to the code:

```
.blue-box {
  border: 3px solid black;
  background-color: skyblue;
}

.orange-box {
  border: 1px solid black;
  margin: 25px 50px;
  text-align: center;
  background-color: orange;
}
```

we will see something like this on the browser:



NOTE: When two elements are stacked vertically, their top and bottom margins are collapsed into a single margin equal to the larger of the two margins. Negative values can be found in margin.

PADDING

The CSS padding properties are used to create space around the content and borders of an element. You can provide margins to the sides separately, just like the border property. The sides' margin properties are:

- padding-top
- padding-right
- padding-bottom
- padding-left

Eg. for the below HTML code:

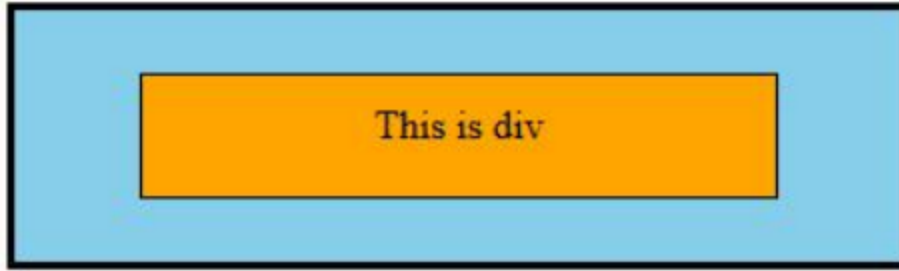
```
<div class="blue-box">
  <div class="orange-box">This is div</div>
</div>
```

and applying the following CSS to the code:

```
.blue-box {
  border: 3px solid black;
  background-color: skyblue;
}

.orange-box {
  border: 1px solid black;
  margin: 25px 50px;
  padding: 10px 0 20px 0;
  text-align: center;
  background-color: orange;
}
```

we will see something like this on the browser:



DISPLAY

This one is a **crucial** CSS property for layout control. It determines **whether or not an element is displayed**.

Many of the elements have default display property values as inline or block (inline and block elements in HTML).

The display property has the following values:

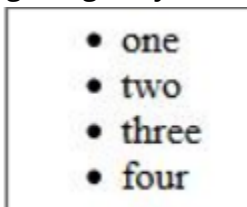
- inline
- block
- inline-block
- none

- Inline Display

When display: inline; is used the following happens to the element:

- The element doesn't start in a new line
- only takes up as much width as necessary, so cannot set height and width
- the vertical margins do not work

Eg., originally a list looks like this:



making the list inline:

```
li { display: inline; }
```

and the list will now be shown in a single line like this(space before the list is because ul element has a default padding value):

one two three four

- Block Display

When ***display: block;*** is used the following happens to the element:

- element starts in a new line
- occupies the entire width of the parent element

Eg., originally the 2 spans look like this:

```
<span>The quick brown fox</span> <span>jumps over the lazy dog!</span>
```

The quick brown fox jumps over the lazy dog!

adding the block property to them like this:

```
li {
  display: block;
}
```

and it will be shown like this now:

The quick brown fox
jumps over the lazy dog!

- Inline-block Display

display: inline-block; is a combination of the properties of the inline and block elements. These are the advantages of inline-block:

- height and width of the element can be set now
- vertical margins are allowed
- The element can sit next to each other

The elements next to each other have a space between them.

Eg., there are two empty div element:

```
<div class="div"></div>
<div class="div"></div>
```

and the following properties have been applied to them:

```
.div {
  display: inline-block;
  border: 1px dashed black;
```

```
width: 100px;
height: 100px;
background-color: lightgrey;
}
```

two divs will be displayed next to each other like this:



You can notice a *space between* these 2 divs. It is because inline elements have **word-spacing in them** and it also defaults in inline-block elements.

- No Display

The `display: none;` property hides the elements in a browser. It actually removes the element from the HTML page and nothing is shown in its place.

This is similar to another property - `visibility: hidden;`. The difference is that the element is not removed from the page and still occupies the space.

POSITION

The **position** property is for specifying the **positioning of elements on the page**. This fixes the position of the element **relative to the web page or parent element**.

The position property provides 5 ways to position the element

- **static** - default
- **relative**
- **absolute**
- **fixed**
- **sticky**

This property is not enough to manipulate the position of the element. The position property just specifies the behavior.

Therefore, to move the elements we have these 4 properties:

- **left** - The element is shifted to the left side of the element as a result of this.
- **right** - The element is shifted to the right side of the element as a result of this.
- **top** - This causes the element to shift in relation to the element's top side.

- **bottom** - This causes the element to shift in relation to the element's bottom side.

The above properties, i.e. **left**, **right**, **top** and **bottom** have numerical values in both relative and absolute units. The value can be both **positive and negative**.

Depending on the position value, they also work differently.

Eg., for the layout like this:

```
<div class="outer">
  <h2>POSITION property</h2>
  <span class="inner">This is relative to viewport</span>
</div>
```

and making div's position relative with css code like:

```
.outer {
  position: relative;
  width: 250px;
  height: 150px;
  border: 1px solid red;
  padding: 10px;
}

.inner {
  border: 1px solid blue;
  padding: 5px;
  top: 10px;
  left: 40px;
}
```

will show on the web page like this:



- **static**

The element is positioned using the **static** value in accordance with the page's normal flow, not in any special way.

This is the **default** property. Properties like a top, right, bottom, left do not work when this is used.

You can alternatively use `position: static;` to apply this property and the layout will look the same.

- relative

The element is positioned **relative** to its **first positioned** (i.e. not static) **ancestor element** by the **relative value**. It's **similar to a static value**, but now the element's properties such as top, right, bottom, and left will work.

The element's original position is still occupied.

Use `position: relative;` to apply this property and the layout will look like this:



- absolute

The **absolute** value positions the element **relative to its closest positioned ancestor**. If there is no positioned ancestor, then the position would be relative to the browser's window.

The element doesn't occupy its original space.

Use `position: absolute;` to apply this property and the layout will look like this:



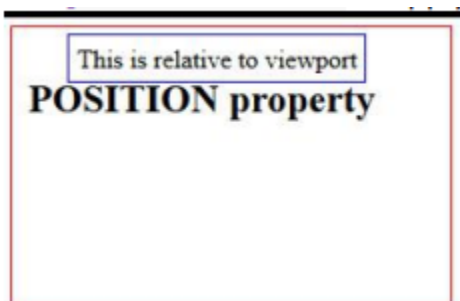
- fixed

The **fixed** value **fixes the position of the element relative to the viewport**, which means it will always stay in the same place, regardless of how far down the page is

scrolled.

In addition, the element does not leave a blank space on the page; it will overlap with another element.

Use `position: fixed;` to apply this property and the layout will look like this:



Notice, the ending of the black line is the start of the web page and the distance between the black line and the is 10px.

- sticky

The **sticky** value is **initially positioned as in the HTML source**. It is then **fixed relative to the viewport as soon as it reaches the desired position**.

Depending on the scroll position, a sticky element switches between relative and fixed states. It moves relative to the viewport until a certain offset is reached, at which point it "sticks" in place (like `position: fixed`).

Use `position: sticky;` to apply this property and the layout initially be same as that of relative positioned and move with the page on scrolling until the final position (i.e. same as fixed position) is reached.

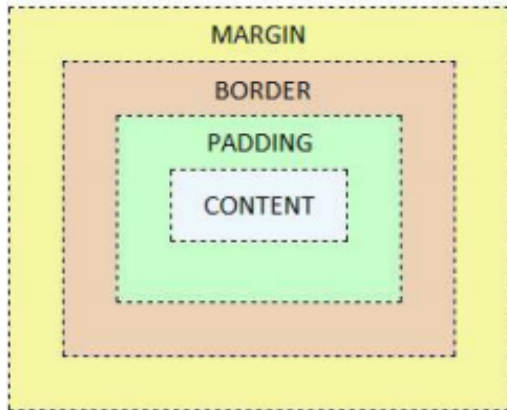
BOX MODEL

Box model is used to specify the layout of the elements. The HTML elements are considered boxes.

Every HTML element is surrounded by a box in the **CSS box model**.

It is made up of the following elements: **margins**, **borders**, **padding**, and **content**.

The image below illustrates the box model:



The **total width** of the element is equal to the total of the horizontal margin, border and padding, and content width of the element.

The **total height** of the element is equal to the total of the horizontal margin, border and padding, and content height of the element.

MIN/MAX WIDTH

The width property is used to set the width of the element, to a specific size. But the size of the becomes fixed with this and this brings the problem of in smaller devices. The browser then gets a scrollbar to scroll through the entire content.

So, to overcome this problem, CSS provides **max-width** property. This specifies the **maximum width that an element can have**. If the browser window's width becomes smaller than the width of the element, the element width adjusts with the browser width.

However, a small element is extremely difficult to read. So, another property **min-width** is provided by the CSS that specifies the minimum width that the element can have.

Eg., we have two divs like this:

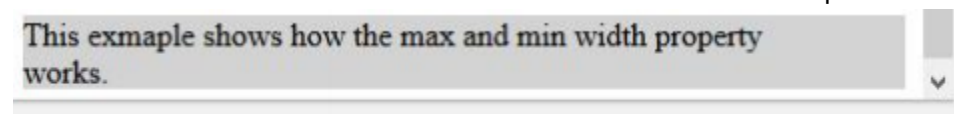
```
<div id="div1">
  <div id="div2">This example shows how the max and min width property works.
</div>
</div>
```

and the CSS is applied to them:

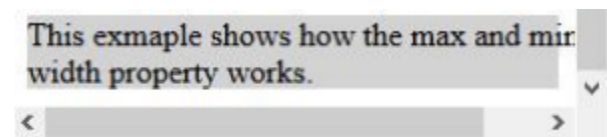
```
#div1 {
  background-color: lightgrey;
}
#div2 {
```

```
max-width: 400px;
min-width: 300px;
}
```

will show like this when the browser width is more than 400px:



and like this when the browser width is less than 300px:



OVERFLOW

The **overflow** property defines **what will happen if the content of an element overflows**, i.e. the height or width of content is larger than the element's height or width. When the content of an element is too large to fit in a specified area, this property adds a scroll bar or clips the content.

The values that the overflow property can take are:

- **visible** - This is the normal setting. The content overflows here and is seen outside the box
- **hidden** - only the content that fits inside the box is visible and the overflow is clipped
- **scroll** - all the content is visible through a scroll-bar added to the box
- **auto** - a scroll-bar gets added if content overflows

Eg., when we a larger content than that can be fitted inside the element like:

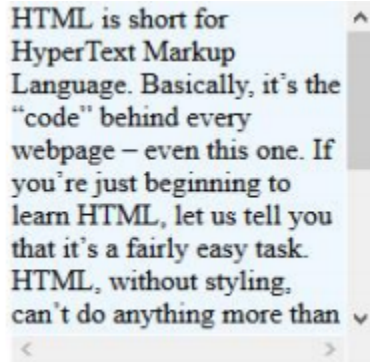
```
<div>HTML is short for HyperText Markup Language. It's basically the "code" that powers every website, including this one. If you're just getting started with HTML, rest assured that it's a relatively simple task. HTML, without styling, can't do anything more than setting a layout, drawing a table, or creating frames – but it is handy as it helps you structure the content correctly, which is important when you sit down to add style to your HTML.</div>
```

with fixed height and width as:

```
div {
  background-color: aliceblue;
  width: 200px;
```

```
height: 200px;
overflow: scroll;
}
```

will show the box in the browser like this:



NOTE: Only block elements with a specified height can use the overflow property.