



Topic : Styling With CSS

1. COLOR

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

The color property can be specified in 6 different ways. Each one of them provides has some difference from the other. Although color can also be applied to 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(64, 213, 240, 0.5)</h2>
<h2 style="color:hsla(9, 100%, 64%, 0.7);">hsla(190, 73%, 95%, 0.5)</h2>
```

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

orangered

#ff6347

rgb(255, 99, 71)

hsl(9, 100%, 64%)

rgba(255, 99, 71, 0.8)

hsla(9, 100%, 64%, 0.8)

1.1. By name

All modern browsers support 140 different colors named in CSS. Unlike HTML, CSS will completely ignore unknown keywords.

The color keywords all represent plain, solid colors, without transparency. Eg., *orangered*, *green*, *blue*, *lightgrey*, etc.

1.2. Using rgb

RGB stands for **Red**, **Green** and **Blue**. It is a color model where **combination of red, green and blue** form a color. The intensity of each color has value ranging from 0 to 255. This provides very large number of colors dataset.

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

1.3. By hex code

The colors can be represented by **6 digits hexadecimal code**. The codes are made using the *3 colors (Red, Green and Blue)*. First 2 digits are red, next 2 are green and last 2 are blue. So, the syntax for hex code is: `#RRGGBB`.

For each hexadecimal value between **00 - FF** is similar to **0 - 255**.

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

1.4. Using hsl

The color can also be specified using the **HSL (Hue, Saturation and Lightness)** components.

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

1.5. Using rgba

RGBA (Red, Green, Blue, Alpha) is an extension of RGB, provided with alpha transparency. This alpha value determines the opacity of the RGB defined color. The alpha parameter is a number between 0.0 (transparent) and 1.0 (opaque).

Eg., `rgba(255, 0, 0, 0.6)` is a red color, with 0.6 opacity will have the color:



1.6. Using hsla

HSLA (Hue, Saturation, Lightness, Alpha) is also an extension of HSL, provided with alpha transparency. The alpha value and property is same as that in RGBA.

Eg., `hsla(0, 100%, 50%, 0.6)` is also a red color with 0.6 opacity and have same color:



2. CSS UNITS

CSS units are used for **expressing size** of different properties.

The units are expressed by a **number followed by the unit symbol**. Many CSS properties take "length" values, such as width, margin, padding, font-size, border-width, etc.

Eg. are 10px, 2em, 50% etc.

A whitespace cannot appear between the number and the unit.
If the value is 0, the unit can be omitted.

For some CSS properties, negative lengths are allowed. Eg. margin
There are two types of length units: absolute and relative.

2.1. Absolute Units

The absolute units a fixed size/length of the element. Absolute length units are not recommended for use on screen, because screen sizes vary so much.

The absolute units consist of the following:

- **cm** - centimeters
- **mm** - millimeters in inches (1in = 96px = 2.54cm)
- **px** - pixels (1px = 1/96th of 1in)
- **pt** - points (1pt = 1/72 of 1in)
- **pc** - picas (1pc = 12 pt)

2.2. Relative Units

Relative length units specify a length relative to another length property.

Some of the relative units are the following:

- **em** - Relative to the font-size of the element (2em means 2 times the size of the current font)
- **rem** - Relative to font-size of the root element
- **vw** - Relative to 1% of the width of the browser window size
- **vh** - Relative to 1% of the height of the browser window size
- **%** - Relative to the parent element

3. BORDER

The **border** property is used to set element's border. CSS borders allows you to specify the **style**, **width**, and **color** of an element's border.

Border is a shorthand for border-width, border-style, and border-color written together.

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

```
border: 5px solid red;
```

will show like:

This is border in CSS

We will look into the different properties of border.

3.1. Border Width

The **border-width** property specifies the width of the four borders. The width can be set as absolute or relative size or by using one of the three predefined values: *thin*, *medium*, or *thick*.

Eg., `border-width: 2px 10px 4px 20px;`, will have top border of 2px, right border of 10px, bottom border to 4px and left border to 20px.

3.2. Border Style

The **border-style** property specifies what kind of border to display. The values for border-style are: *dotted*, *dashed*, *solid*, *double*, *groove*, *ridge*, *inset*, *outset*, *none*, *hidden*.

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.*

3.3. Border Color

The **border-color** property specifies the color of the four borders. The value of the property is same as that of the color property. But now you can provide different color to different border sides. If border-color is not set, it inherits the color of the element.

Eg., `border-color: red green;`, will have top and bottom border color red and left and right border color green.

3.4. Border Individual Sides

Using the border property, we are able to provide width, style and color to all the borders separately but still we have to give some value to all the sides of the border.

CSS border also have property to give border value separately to each of the border sides. The border property for the sides are:

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

This is further broken to provide style, width and color separately to the border sides. Some of them are: **border-top-style**, **border-right-width**, **border-left-color**, etc.

3.5. Border Radius

The **border-radius** property is used to add rounded borders to an element. You can give absolute(eg. in px) or relative(eg. in %) value to this property.

Eg., if we add `border-radius: 10px;` property to the first border example, will show like:



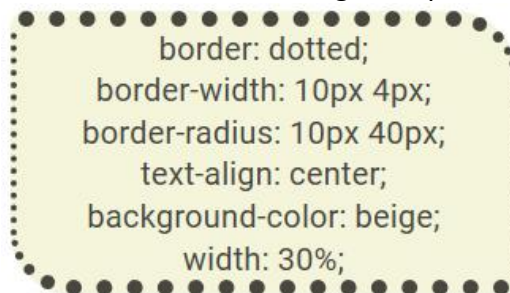
This is border in CSS

The border-radius can be provided in **elliptical form** as well. Therefore, you need to provide horizontal and vertical radius differently. This is done using a slash ("/") between horizontal and vertical radius. Here is an example:

```
div {  
  border-radius: 30px/10px; /* horizontal radius / vertical radius */  
  background: teal;  
  width:20%;  
  height:20%;  
  padding:40px;  
}
```



Here is another interesting example:



4. TEXT AND FONT STYLING

Various properties are provided to change the look and style of text in the HTML document. These styles will apply only to content of any element that is text. We will look into some of the mostly used text and font styling properties.

Font properties define the look of the font like font family, boldness, size and style. The first 4 are font properties.

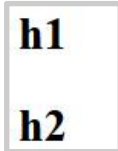
Text properties define the layout and presentation of the text in the HTML page.

4.1. font-size

This defines the **size of the text**. The **font-size** value can be an absolute, or relative size, i.e., values can be applied in px, %, em, etc.

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

will show



4.2. font-family

The font family of a text is set with the **font-family** property.

The font-family property should hold several font names as a "**fallback**" system. If the browser does not support the first font, it tries the next font, and so on.

Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

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

will show



NOTE: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".

4.3. font-weight

The **font-weight** property specifies the **weight/thickness of a font**. The weight ranges from light to bold. Values can be bold, bolder, inherit, initial, lighter, normal, unset. Alternatively, we can use numeric values ranging from 100-900 to define the weight of the font.

Eg., `h1 { font-weight: lighter; }`

will show



4.4. font-style

The **font-style** property specifies the **style for a text**. The values for this property are: **normal, italic, oblique, initial, inherit**.

4.5. 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.

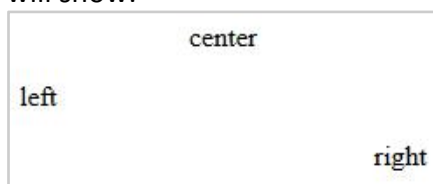
4.6. text-align

The **text-align** property is used to **set 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:



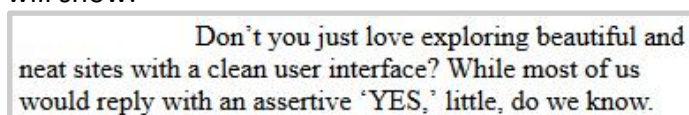
4.7. text-indent

The **text-indent** property specifies the **indentation of the first line in a text-block**. **Negative values are allowed**. The first line will be indented to the left if the value is negative.

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:



4.8. text-transform

The **text-transform** property is used to specify the **case of the letters in a text**.

It can be used to turn text to:

- **uppercase** - turns every character to uppercase
- **lowercase** - turns every character to lowercase
- **capitalize** - turns first letter of each word to uppercase and other to lowercase
- **none** - text renders as it is. It is the default

4.9. text-decoration

The **text-decoration** property is used to **set or remove decorations from text**.

The value **text-decoration: none;** is often used to **remove underlines from links**.

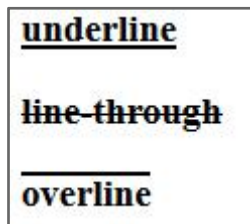
The text-decoration is used to decorate text. It has 4 values:

- **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:



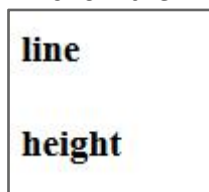
4.10. line-height

The **line-height** property sets the **height of the lines in an element**. It does not change the size of the font.

The font-size value can be an absolute, or relative size, i.e., values can be applied in px, %, em, etc. If no unit specified then it is multiplied by element's font-size.

Eg., `h3 { line-height: 1.5; }`

will show the lines like:



4.11. letter-spacing

The **letter-spacing** property is used to specify the **space between the characters** in a text. This is used to increase or decrease the space between the characters only. The value of letter-spacing can be absolute or relative.

Eg., `h3 { letter-spacing: 0.5em; }`
will show the text like:

l e t t e r s p a c i n g

4.12. word-spacing

The **word-spacing** property is used to specify the **space between the words** in a text. This is used 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

4.13. text-shadow

The **text-shadow** property **adds shadow to text**. The value contains 3 values, which are **position of the horizontal shadow**, the **position of the vertical shadow** and the **color of the shadow**.

Eg., `h2 { text-shadow: 2px 1px red; }`
will show the text like:

text shadow

5. BACKGROUND

The background properties are used to **define the background effects for elements**. The background of an element is the total size of the element and includes padding and border but not the margin.

Backgrounds can be filled with a color or image, clipped or resized, and otherwise modified.

CSS background properties:

- background-image
- background-repeat
- background-position
- background-size

- background-attachment

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

```
body {
  background-image: url("https://blog.codingninjas.in/wp-
    content/uploads/2017/01/cropped-Final_logo_switchtocode-01.png"), linear-
    gradient(#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>

5.1. background-color

The **background-color** property sets the **background color of an element**. It has the same value 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:

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.

5.2. background-image

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

This can set one or more background images for an element.

By default, a background-image is placed at the top-left corner of an element, and is repeated so it covers the entire element both vertically and horizontally.

The values it can take are:

- **url('URL')** - specifies the URL of the image. You can specify more than one image by separating the URLs with a comma
- **none** - this is **default** value. No background image will be displayed
- **linear-gradient()** - sets a linear gradient as the background image. At least two colors needed to be mentioned (default direction is top to bottom)
- **radial-gradient()** - sets a radial gradient as the background image. At least two colors needed to be mentioned (default is from center to edges)
- **repeating-linear-gradient()** - repeats a linear gradient
- **repeating-radial-gradient()** - repeats a radial gradient

In the above example, we are using both the image and linear-gradient together. Try to swap the sequence of image and gradient and see what happens.

5.3. background-repeat

The **background-repeat** property is used to **specify how/if a background image will be repeated**.

By default, a background image repeats both vertically and horizontally, so background-repeat will how will the image repetition works.

The values this property can take are:

- **repeat** - this is default value. The background image is repeated both vertically and horizontally. The last image will be clipped if it does not fit
- **repeat-x** - the image is repeated only horizontally
- **repeat-y** - the image is repeated only vertically
- **no-repeat** - the image will only be shown once
- **space** - the background-image is repeated without clipping. The space remaining is distributed evenly between images with first and last images pinned to sides of the element
- **round** - the image is repeated and shrink or stretch to fill the space

5.4. background-position

The **background-position** property is used to **specify the initial position of a background image**.

By default, a background image is placed at the top-left corner of an element and you can change the position with background-position property.

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 one value is specified, the other value will be "center"
- **Xpos Ypos** - specifies the horizontal and vertical position relative to the viewport. Units can be any of the CSS units. If you only specify one value, the other value will be 50%.

5.5. background-size

The **background-size** property is used to **specify the size of the background images**.

The values it can take are:

- **auto** - this is a default value. The image is displayed in its original size
- **length** - sets the width and height of the background image. The first value sets the width, the second value sets the height
- **percentage** - sets the width and height of the background image in percent. The first value sets the width, the second value sets the height. If only one value is given, the second is set to "auto"
- **cover** - resizes the background image to cover horizontal width of container
- **contain** - resizes the background image to make sure the image is fully visible

5.6. background-attachment



The **background-attachment** property is used to **specify whether a background image scrolls with the rest of the page, or is fixed**.

The values it can take are:

- **scroll** - this is default value. The background image will scroll with the page
- **fixed** - the background image will not scroll with the page
- **local** - the background image will scroll with the element's contents

6. 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-top
- margin-right
- margin-bottom
- margin-left

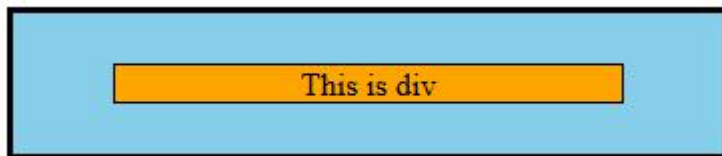
Eg. for the below HTML code:

```
<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;  
  text-align: center;  
  background-color: orange;  
}
```

we will see something like this on the browser:



NOTE: When two elements are positioned vertically, then the top and bottom margins of elements are collapsed into a single margin that is equal to the largest of the two margins. Margin can contain negative values.

7. PADDING

The CSS padding properties are used to generate **space around an element's content, and its borders**.

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

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

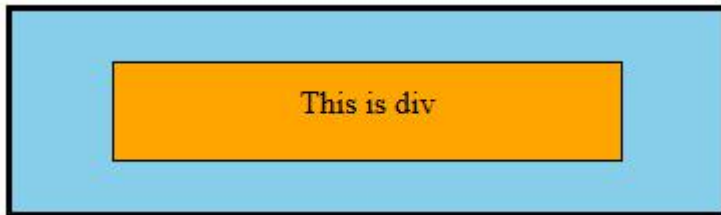
Eg. for the below HTML code:

```
<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:



8. DISPLAY

The display property is a very important CSS **property for controlling layout**. It specifies **if/how an element is displayed**.

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

The display property has following values:

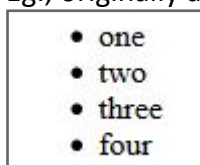
- inline
- block
- inline-block
- none

8.1. Inline Display

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

- 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

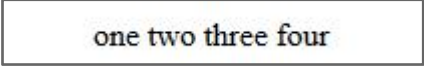
Eg., originally a list looks like this:



making the list inline:

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

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



one two three four

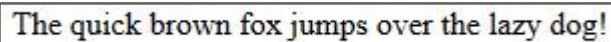
8.2. Block Display

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

- element starts in a new line
- takes up full-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>
```

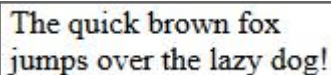


The quick brown fox jumps over the lazy dog!

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!

8.3. 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
- element can sit next to each other

There is a space between the elements next to each other

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 a **word-spacing in them** and it is also a default in inline-block elements.

8.4. 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.

9. POSITION

The **position** property is for specifying the **positioning of element on the page**. This fixes the position of element **relative to 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 behaviour. Therefore, to move the elements we have these 4 properties:

- **left** - shifts the element with respect to the left side of the element
- **right** - shifts the element with respect to the right side of the element
- **top** - shifts the element with respect to the top side of the element
- **bottom** - shifts the element with respect to the bottom side of the element

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**.

They also *work differently depending on the position value*.

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:



9.1. static

The **static** value positions the element according to the **normal flow of the page** and not in any special way. This is the **default** property. Properties like 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 same.

9.2. relative

The **relative** value positions the element **relative to its first positioned (i.e. not static) ancestor element**. It is similar to the static value, but now the properties like top, right, bottom, left will work on the element.

The original position of the element remains occupied.

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



9.3. 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 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:

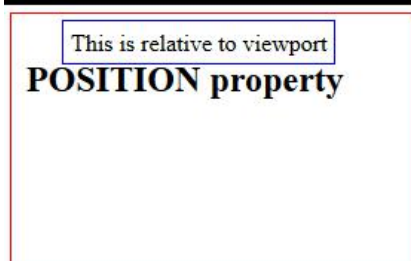


9.4. fixed

The **fixed** value **fixes the position of the element relative to the viewport**, which means it always stays in the same place even if the page is scrolled.

Also, **the element does not leave a gap in the page**, i.e. it will overlap with other element.

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



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

9.5. 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**.

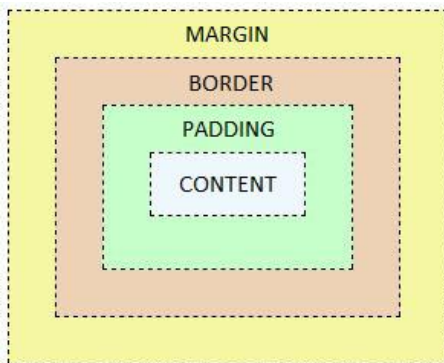
A sticky element **toggles between relative and fixed**, depending on the scroll position. It is positioned relative until a given offset position is met in the viewport - then it "sticks" in place (like position:fixed).

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

10. BOX MODEL

Box model describes the layout of the elements. The HTML elements are considered as boxes.

The CSS box model is essentially a **box that wraps around every HTML element**. It consists of: **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 element.

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

11. MIN/MAX WIDTH

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 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.

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

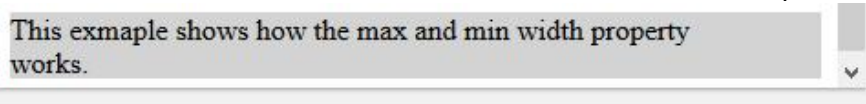
Eg., we have two divs like this:

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

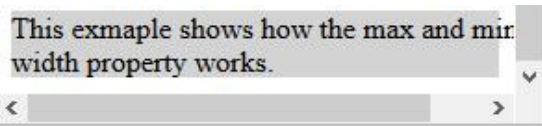
and the CSS is applied to them:

```
#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:



12. OVERFLOW

The **overflow** property is used to **specify what happens if the content of an element overflows**, i.e. the content's height or width is larger than element's height or width. This property adds a scroll-bar or clips the content when an element's content is too big to fit in a specified area.

The values that the overflow property can take are:

- **visible** -this is **default value**. The content overflows and 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:

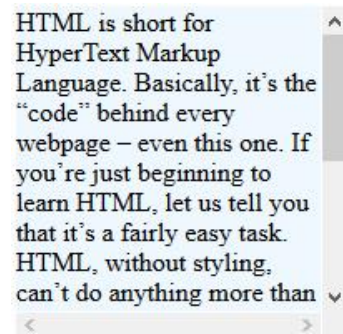
```
<div>HTML is short for HyperText Markup Language. Basically, it's the "code" behind
every webpage - even this one. If you're just beginning to learn HTML, let us tell
you that it's a fairly easy task. HTML, without styling, can't do anything more
than setting a layout, drawing a table, or creating 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: The *overflow* property only works for block elements with a specified height.