CSS Cascading Style Sheets 6

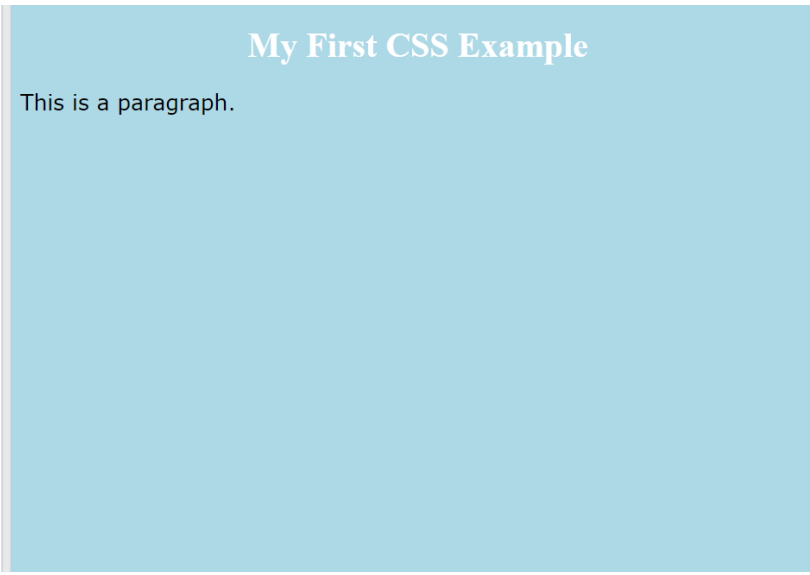
CSS Cascading Style Sheets

- CSS stands for Cascading Style Sheets
- CSS describes how HTML elements are to be displayed on screen, paper, or in other media
- 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

Why Use CSS?

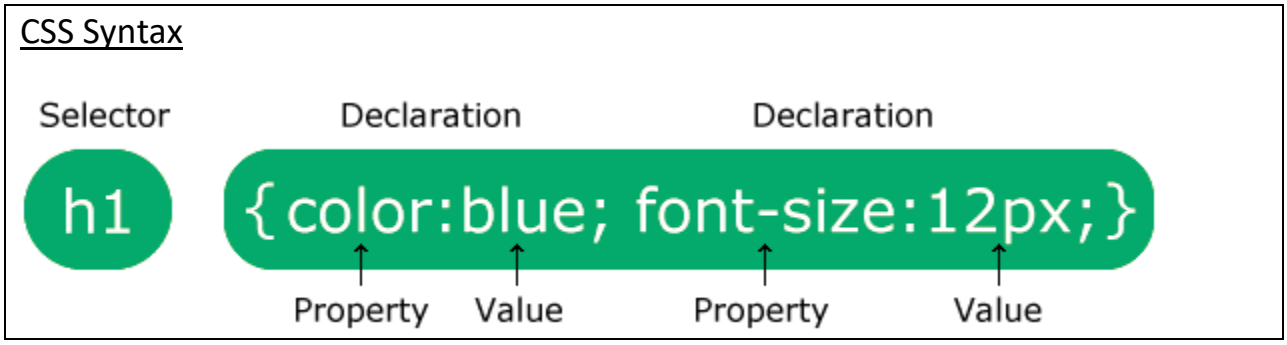
CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes.

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

h1 {
  color: white;
  text-align: center;
}

p {
  font-family: verdana;
  font-size: 20px;
}
</style>
</head>
<body>

<h1>My First CSS Example</h1>
<p>This is a paragraph.</p>

</body>
</html>
```



CSS Syntax

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



The selector points to the HTML element you want to style.

The declaration block contains one or more declarations separated by semicolons.

Each declaration includes a CSS property name and a value, separated by a colon.

Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

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

<p>Hello World!</p>
<p>These paragraphs are styled with CSS.</p>

</body>
</html>
```

Hello World!

These paragraphs are styled with CSS.

Add CSS in html

There are three ways of inserting a style sheet:

- External CSS
- Internal CSS
- Inline CSS

External CSS

With an external style sheet, you can change the look of an entire website by changing just one file!

Each HTML page must include a reference to the external style sheet file inside the <link> element, inside the head section.

External styles are defined within the <link> element, inside the <head> section of an HTML page:

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

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

An external style sheet can be written in any text editor, and must be saved with a .css extension.

The external .css file should not contain any HTML tags.

Here is how the "mystyle.css" file looks:

"mystyle.css"

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

h1 {
  color: navy;
  margin-left: 20px;
}
```

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. Internal styles are defined within the <style> element, inside the <head> section of an HTML page:

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

h1 {
  color: maroon;
  margin-left: 40px;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

Inline CSS

An inline style may be used to apply a unique style for a single element.

To use inline styles, add the style attribute to the relevant element. The style attribute can contain any CSS property.

Inline styles are defined within the "style" attribute of the relevant element:

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

<h1 style="color:blue;text-align:center;">This is a heading</h1>
<p style="color:red;">This is a paragraph.</p>

</body>
</html>
```

Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

Assume that an **external style sheet** has the following style for the <h1> element:

```
h1 {
  color: navy;
}
```

Then, assume that an **internal style sheet** also has the following style for the <h1> element:

```
h1 {
  color: orange;
}
```

If the internal style is defined **after** the link to the external style sheet, the <h1> elements will be "orange":

```
<head>
<link rel="stylesheet" type="text/css" href="mystyle.css">
<style>
h1 {
  color: orange;
}
</style>
</head>
```

However, if the internal style is defined **before** the link to the external style sheet, the <h1> elements will be "navy":

```
<head>
<style>
h1 {
  color: orange;
}
</style>
<link rel="stylesheet" type="text/css" href="mystyle.css">
</head>
```

Selectors

CSS selectors are used to "find" (or select) the HTML elements you want to style.

We can divide CSS selectors into five categories:

- Simple selectors (select elements based on name, id, class)
- Combinator selectors (select elements based on a specific relationship between them)
- Pseudo-class selectors (select elements based on a certain state)
- Pseudo-elements selectors (select and style a part of an element)
- Attribute selectors (select elements based on an attribute or attribute value)

Element Selector

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

Here, all <p> elements on the page will be center-aligned, with a red text color:

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

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

<p>Every paragraph will be affected by the style.</p>
<p id="para1">Me too!</p>
<p>And me!</p>

</body>
</html>
```

Every paragraph will be affected by the style.

Me too!

And me!

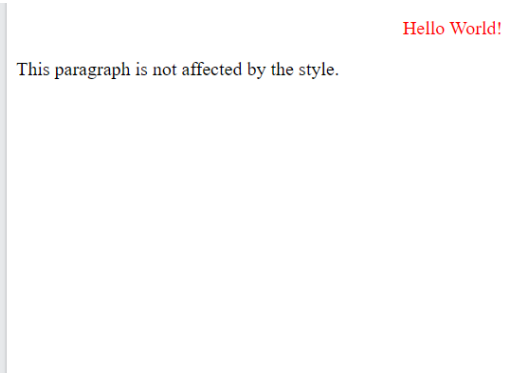
id Selector

The id selector uses the id attribute of an HTML element to select a specific element.
The id of an element is unique within a page, so the id selector is used to select one unique element!
To select an element with a specific id, write a hash (#) character, followed by the id of the element.

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

```
#para1 {
  text-align: center;
  color: red;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
#para1 {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<p id="para1">Hello World!</p>
<p>This paragraph is not affected by the style.</p>

</body>
</html>
```



class Selector

The class selector selects HTML elements with a specific class attribute.

To select elements with a specific class, write a period (.) character, followed by the class name.

In this example all HTML elements with class="center" will be red and center-aligned:

```
.center {
  text-align: center;
  color: red;
}
```

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

<h1 class="center">Red and center-aligned heading</h1>
<p class="center">Red and center-aligned paragraph.</p>

</body>
</html>
```

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

In this example only <p> elements with class="center" will be red and center-aligned:

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

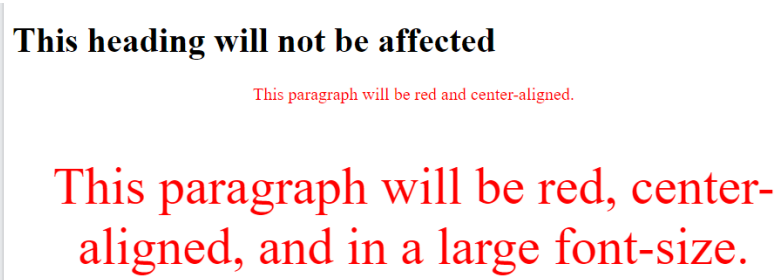
```
<!DOCTYPE html>
<html>
<head>
<style>
p.center {
  text-align: center;
  color: red;
}
</style>
</head>
<body>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>

</body>
</html>
```

HTML elements can also refer to more than one class.

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

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

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

p.large {
  font-size: 300%;
}
</style>
</head>
<body>

<h1 class="center">This heading will not be affected</h1>
<p class="center">This paragraph will be red and center-aligned.</p>
<p class="center large">This paragraph will be red, center-aligned, and in a large font-size.</p>

</body>
</html>
```



Universal Selector

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

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

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

Grouping Selector

The grouping selector selects all the HTML elements with the same style definitions.

Look at the following CSS code (the h1, h2, and p elements have the same style definitions):

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

h2 {
  text-align: center;
  color: red;
}

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

It will be better to group the selectors, to minimize the code.

To group selectors, separate each selector with a comma.

In this example we have grouped the selectors from the code above:

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

CSS Comments

```
/* This is a single-line comment */
```

Colors

Colors are specified using predefined color names, or RGB, HEX, HSL, RGBA, HSLA values.

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

<h1 style="color:tomato;">Tomato</h1>
<h1 style="background-color:Orange;">Orange</h1>
<h1 style="background-color:DodgerBlue;">DodgerBlue</h1>
<h1 style="background-color:MediumSeaGreen;">MediumSeaGreen</h1>
<h1 style="background-color:Gray;">Gray</h1>
<h1 style="background-color:SlateBlue;">SlateBlue</h1>
<h1 style="background-color:Violet;">Violet</h1>
<h1 style="background-color:LightGray;">LightGray</h1>

</body>
</html>
```

Tomato

Orange

DodgerBlue

MediumSeaGreen

Gray

SlateBlue

Violet

LightGray

border color

CSS Border Color

You can set the color of borders:

Hello World

Hello World

Hello World

Example

```
<h1 style="border:2px solid Tomato;">Hello World</h1>
<h1 style="border:2px solid DodgerBlue;">Hello World</h1>
<h1 style="border:2px solid Violet;">Hello World</h1>
```

Color Values

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

Same as color name "Tomato":

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)

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

<p>Same as color name "Tomato":</p>

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

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

<p>In addition to the predefined color names, colors can be specified using RGB, HEX,
HSL, or even transparent colors using RGBA or HSLA color values.</p>

</body>
</html>
```

Same as color name "Tomato":

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)

In addition to the predefined color names, colors can be specified using RGB, HEX, HSL, or even transparent colors using RGBA or HSLA color values.

RGB Colors

In CSS, a color can be specified as an RGB value, using this formula:

rgb(*red, green, blue*)

Each parameter (red, green, and blue) defines the intensity of the color between 0 and 255.

For example, rgb(255, 0, 0) is displayed as red, because red is set to its highest value (255) and the others are set to 0.

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).

Shades of gray are often defined using equal values for all the 3 light sources:

Example

RGBA Value (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.

An RGBA color value is specified with:

rgba(*red, green, blue, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

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

<h1>Make transparent colors with RGBA</h1>

<h2 style="background-color:rgba(255, 99, 71, 0);">rgba(255, 99, 71, 0)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.2);">rgba(255, 99, 71, 0.2)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.4);">rgba(255, 99, 71, 0.4)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.6);">rgba(255, 99, 71, 0.6)</h2>
<h2 style="background-color:rgba(255, 99, 71, 0.8);">rgba(255, 99, 71, 0.8)</h2>
<h2 style="background-color:rgba(255, 99, 71, 1);">rgba(255, 99, 71, 1)</h2>

</body>
</html>
```

Make transparent colors with RGBA

rgba(255, 99, 71, 0)

rgba(255, 99, 71, 0.2)

rgba(255, 99, 71, 0.4)

rgba(255, 99, 71, 0.6)

rgba(255, 99, 71, 0.8)

rgba(255, 99, 71, 1)

HEX Colors (hexadecimal)

A hexadecimal color is specified with: #RRGGBB, where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.

In CSS, a color can be specified using a hexadecimal value in the form:

#rrggbb

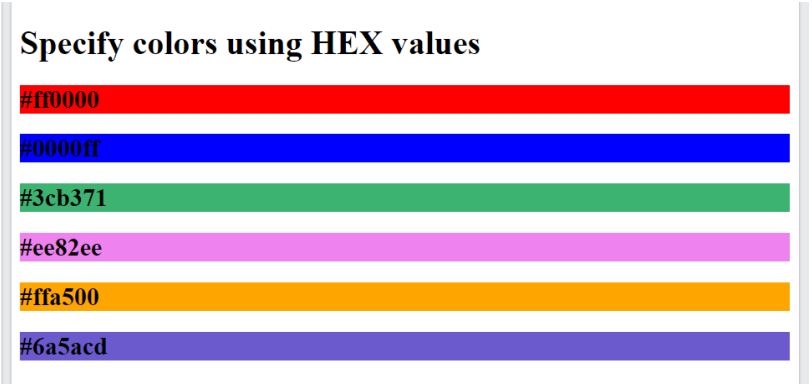
Where rr (red), gg (green) and bb (blue) are hexadecimal values between 00 and ff (same as decimal 0-255).

For example, #ff0000 is displayed as red, because red is set to its highest value (ff) and the others are set to the lowest value (00).

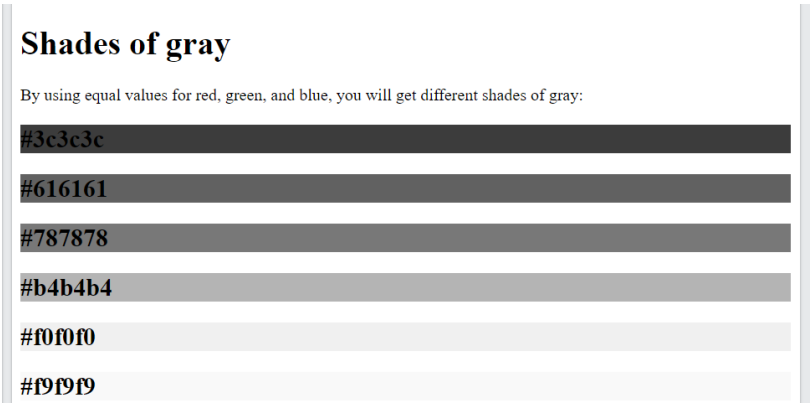
To display black, set all values to 00, like this: #000000.

To display white, set all values to ff, like this: #ffffff.

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

<h1>Specify colors using HEX values</h1>

<h2 style="background-color:#ff0000;">#ff0000</h2>
<h2 style="background-color:#0000ff;">#0000ff</h2>
<h2 style="background-color:#3cb371;">#3cb371</h2>
<h2 style="background-color:#ee82ee;">#ee82ee</h2>
<h2 style="background-color:#ffa500;">#ffa500</h2>
<h2 style="background-color:#6a5acd;">#6a5acd</h2>

</body>
</html>
```




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

<h1>Shades of gray</h1>

<p>By using equal values for red, green, and blue, you will get different shades of gray:
</p>

<h2 style="background-color:#3c3c3c;">#3c3c3c</h2>
<h2 style="background-color:#616161;">#616161</h2>
<h2 style="background-color:#787878;">#787878</h2>
<h2 style="background-color:#b4b4b4;">#b4b4b4</h2>
<h2 style="background-color:#f0f0f0;">#f0f0f0</h2>
<h2 style="background-color:#f9f9f9;">#f9f9f9</h2>

</body>
</html>
```



3 Digit HEX Value

Sometimes you will see a 3-digit hex code in the CSS source.
The 3-digit hex code is a shorthand for some 6-digit hex codes.
The 3-digit hex code has the following form:

#rgb

Where r, g, and b represent the red, green, and blue components with values between 0 and f.
The 3-digit hex code can only be used when both the values (RR, GG, and BB) are the same for each component. So, if we have #ff00cc, it can be written like this: #f0c.

```css
body {
  background-color: #fc9; /* same as #ffcc99 */
}

h1 {
  color: #f0f; /* same as #ff00ff */
}

p {
  color: #b58; /* same as #bb5588 */
}
```

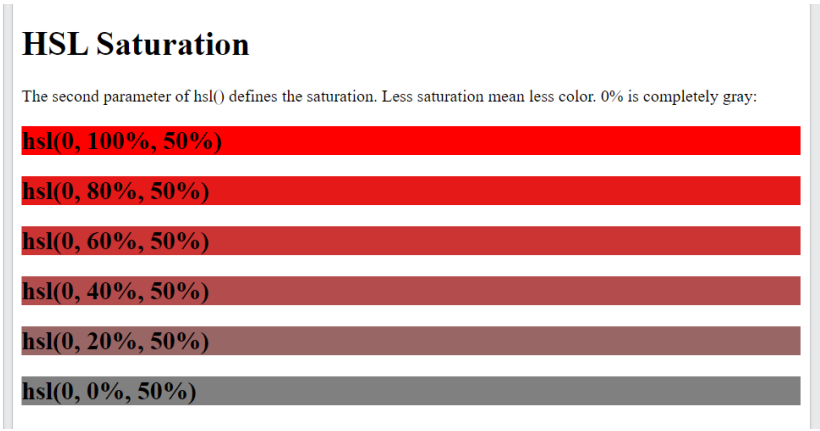
HSL Colors (hue, saturation, lightness)

HSL stands for hue, saturation, and lightness.
In CSS, a color can be specified using hue, saturation, and lightness (HSL) in the form:

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

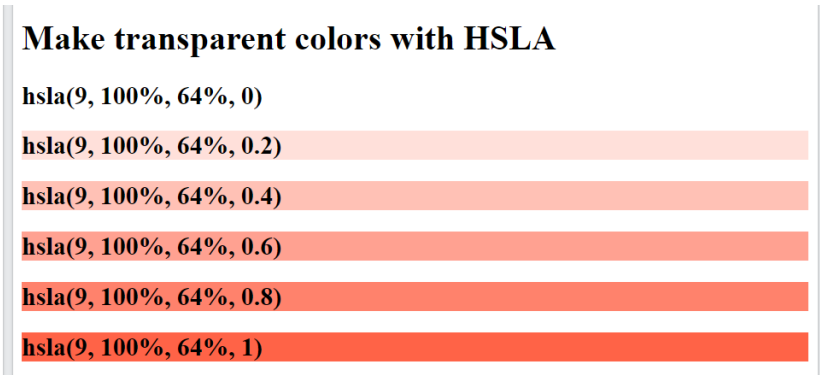
```
<!DOCTYPE html>
<html>
<body>

<h1>HSL Saturation</h1>

<p>The second parameter of hsl() defines the saturation. Less saturation mean less color.
0% is completely gray:</p>

<h2 style="background-color:hsl(0, 100%, 50%);">hsl(0, 100%, 50%)</h2>
<h2 style="background-color:hsl(0, 80%, 50%);">hsl(0, 80%, 50%)</h2>
<h2 style="background-color:hsl(0, 60%, 50%);">hsl(0, 60%, 50%)</h2>
<h2 style="background-color:hsl(0, 40%, 50%);">hsl(0, 40%, 50%)</h2>
<h2 style="background-color:hsl(0, 20%, 50%);">hsl(0, 20%, 50%)</h2>
<h2 style="background-color:hsl(0, 0%, 50%);">hsl(0, 0%, 50%)</h2>

</body>
</html>
```



hsla(*hue, saturation, lightness, alpha*)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (not transparent at all):

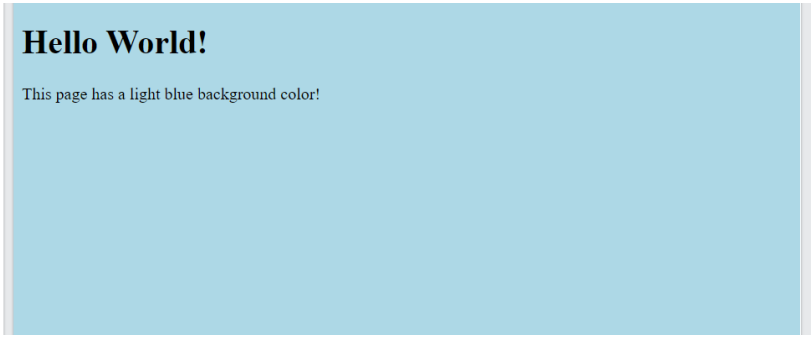
```
<!DOCTYPE html>
<html>
<body>

<h1>Make transparent colors with HSLA</h1>

<h2 style="background-color:hsla(9, 100%, 64%, 0);">hsla(9, 100%, 64%, 0)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.2);">hsla(9, 100%, 64%, 0.2)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.4);">hsla(9, 100%, 64%, 0.4)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.6);">hsla(9, 100%, 64%, 0.6)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 0.8);">hsla(9, 100%, 64%, 0.8)</h2>
<h2 style="background-color:hsla(9, 100%, 64%, 1);">hsla(9, 100%, 64%, 1)</h2>

</body>
</html>
```



Backgrounds COLOR

The `background-color` property specifies the background color of an element.

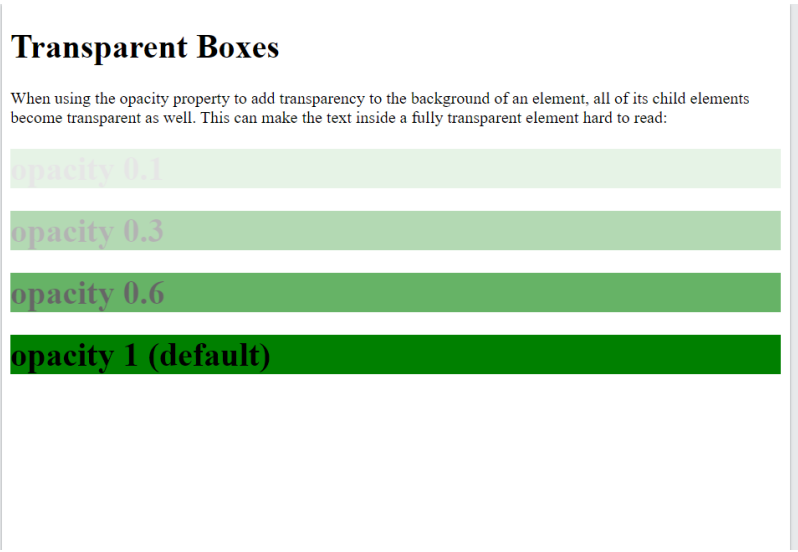
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-color: lightblue;
}
</style>
</head>
<body>

<h1>Hello World!</h1>

<p>This page has a light blue background color!</p>

</body>
</html>
```



Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element.
It can take a value from 0.0 - 1.0. The lower value, the more transparent:

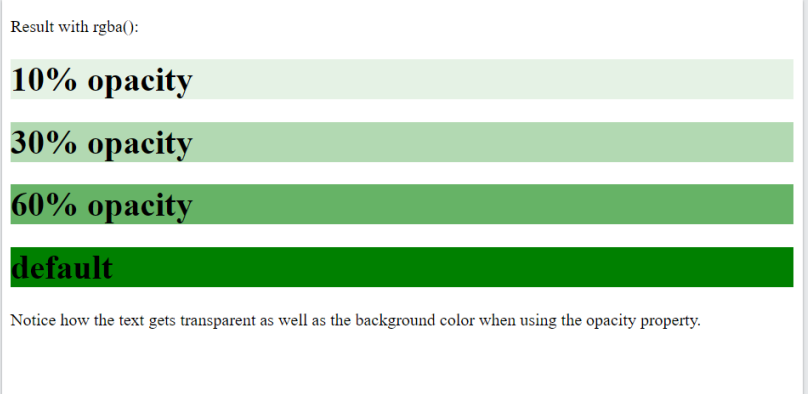
```
<!DOCTYPE html>
<html>
<head>
<style>
div {background-color: green;}
div.first {opacity: 0.1;} div.second {opacity: 0.3;} div.third {opacity: 0.6;}</style>
</head>
<body>
<h1>Transparent Boxes</h1>
<p>When using the opacity property to add transparency to the background of an element,
all of its child elements become transparent as well. This can make the text inside a
fully transparent element hard to read:</p>
<div class="first">
  <h1>opacity 0.1</h1>
</div>
<div class="second">
  <h1>opacity 0.3</h1>
</div>

<div class="third">
  <h1>opacity 0.6</h1>
</div>

<div>
  <h1>opacity 1 (default)</h1>
</div>

</body>
</html>
```



Transparency using RGBA

If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values. The following example sets the opacity for the background color and not the text:

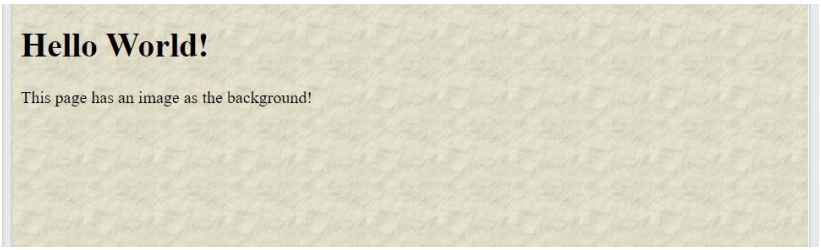
```
<!DOCTYPE html>
<html>
<head>
<style>
div {background: rgb(0, 128, 0);}
div.first {background: rgba(0, 128, 0, 0.1);}
div.second {background: rgba(0, 128, 0, 0.3);}
div.third {background: rgba(0, 128, 0, 0.6);}
</style>
</head>
<body>
<p>Result with rgba():</p>
<div class="first"><h1>10% opacity</h1></div>
<div class="second"><h1>30% opacity</h1></div>
<div class="third"><h1>60% opacity</h1></div>
<div><h1>default</h1></div>
<p>Notice how the text gets transparent as well as the background color when using the
opacity property.</p>
</body>
</html>
```



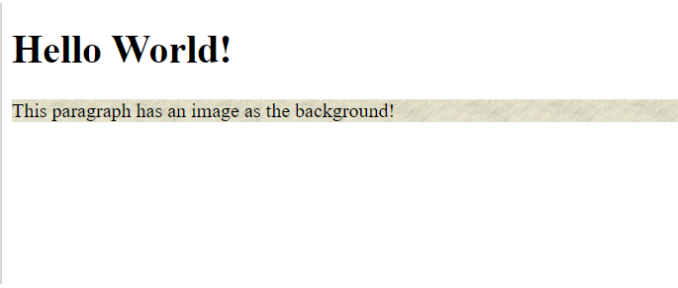
background-image

The `background-image` property specifies an image to use as the background of an element.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {background-image: url("paper.gif");}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>This page has an image as the background!</p>
</body>
</html>
```




```
<!DOCTYPE html>
<html>
<head>
<style>
p {background-image: url("paper.gif");}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>This paragraph has an image as the background!</p>
</body>
</html>
```



background-repeat

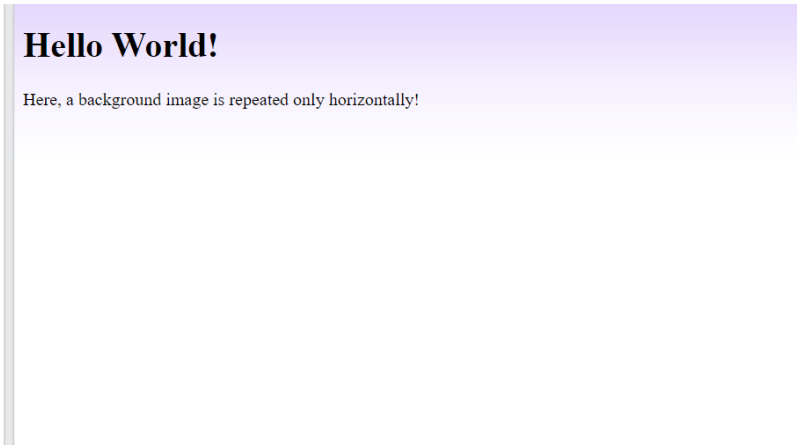
By default, the `background-image` property repeats an image both horizontally and vertically.

Some images should be repeated only horizontally or vertically, or they will look strange, like this:

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

<h1>Hello World!</h1>
<p>Strange background image...</p>

</body>
</html>
```

If the image above is repeated only horizontally (`background-repeat: repeat-x;`), the background will look better:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("gradient_bg.png");
  background-repeat: repeat-x;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>Here, a background image is repeated only horizontally!</p>
</body>
</html>
```

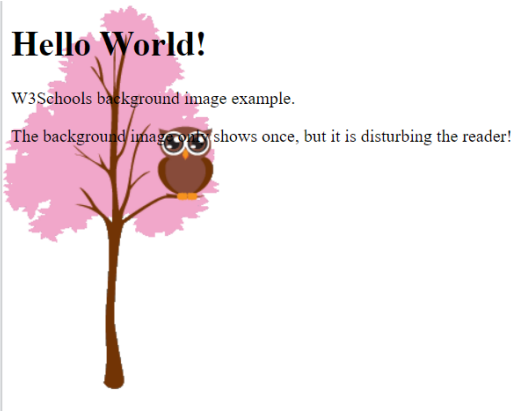


Tip: To repeat an image vertically, set `background-repeat: repeat-y;`

background-repeat: no-repeat

Showing the background image only once is also specified by the `background-repeat` property:

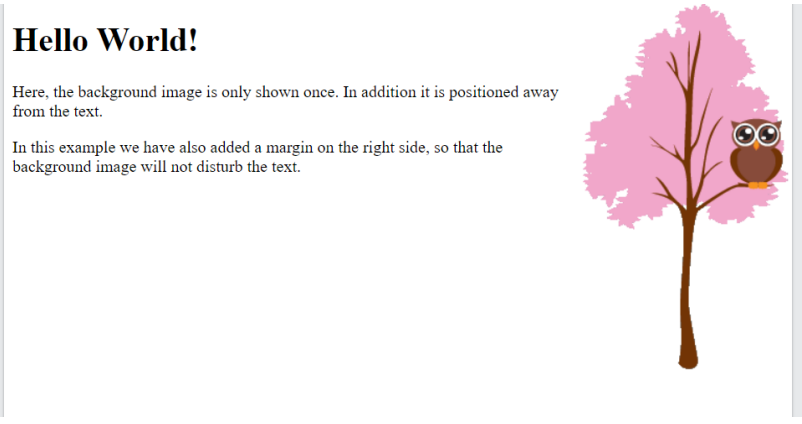
```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
</style>
</head>
<body>
<h1>Hello World!</h1>
<p>W3Schools background image example.</p>
<p>The background image only shows once, but it is disturbing the reader!</p>
</body>
</html>
```



background-position

The `background-position` property is used to specify the position of the background image.

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  margin-right: 200px;
}
</style>
</head>
<body>

<h1>Hello World!</h1>
<p>Here, the background image is only shown once. In addition it is positioned away from the text.</p>
<p>In this example we have also added a margin on the right side, so that the background image will not disturb the text.</p>

</body>
</html>
```

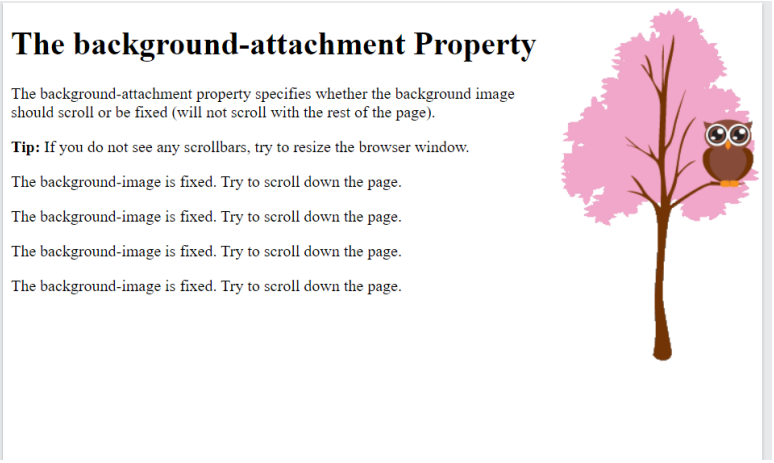


background-attachment

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

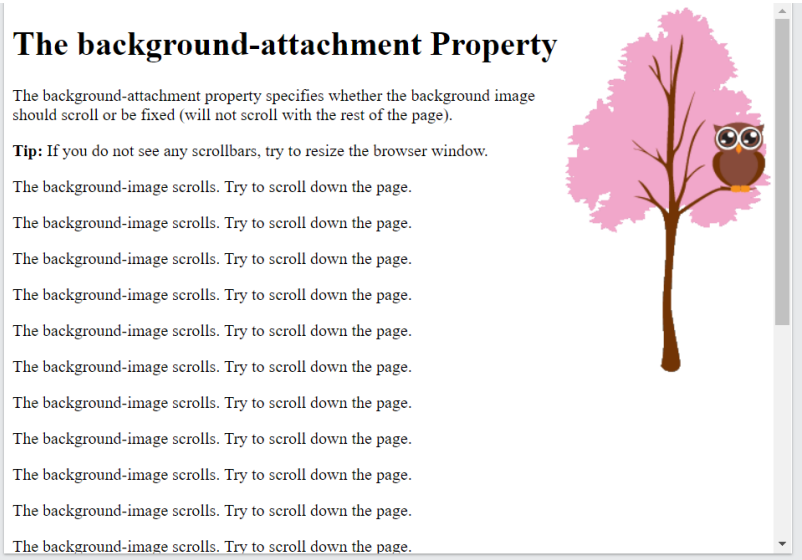
Specify that the background image should be fixed:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  margin-right: 200px;
  background-attachment: fixed;
}
</style>
</head>
<body>
<h1>The background-attachment Property</h1>
<p>The background-attachment property specifies whether the background image should
scroll or be fixed (will not scroll with the rest of the page).</p>
<p><strong>Tip:</strong> If you do not see any scrollbars, try to resize the browser
window.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
<p>The background-image is fixed. Try to scroll down the page.</p>
</body>
</html>
```



Specify that the background image should scroll with the rest of the page:

```
<!DOCTYPE html>
<html>
<head>
<style>
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
  margin-right: 200px;
  background-attachment: scroll;
}
</style>
</head>
<body>

<h1>The background-attachment Property</h1>

<p>The background-attachment property specifies whether the background image should
scroll or be fixed (will not scroll with the rest of the page).</p>

<p><strong>Tip:</strong> If you do not see any scrollbars, try to resize the browser
window.</p>

<p>The background-image scrolls. Try to scroll down the page.</p>
<p>The background-image scrolls. Try to scroll down the page.</p>
<p>The background-image scrolls. Try to scroll down the page.</p>
<p>The background-image scrolls. Try to scroll down the page.</p>
<p>The background-image scrolls. Try to scroll down the page.</p>
<p>The background-image scrolls. Try to scroll down the page.</p>
```



background - Shorthand property

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

Instead of writing:

```
body {
  background-color: #ffffff;
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

You can use the shorthand property `background`:

Use the shorthand property to set the background properties in one declaration:

```
body {
  background: #ffffff url("img_tree.png") no-repeat right top;
}
```

When using the shorthand property the order of the property values is:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position

It does not matter if one of the property values is missing, as long as the other ones are in this order. Note that we do not use the background-attachment property in the examples above, as it does not have a value.

Borders

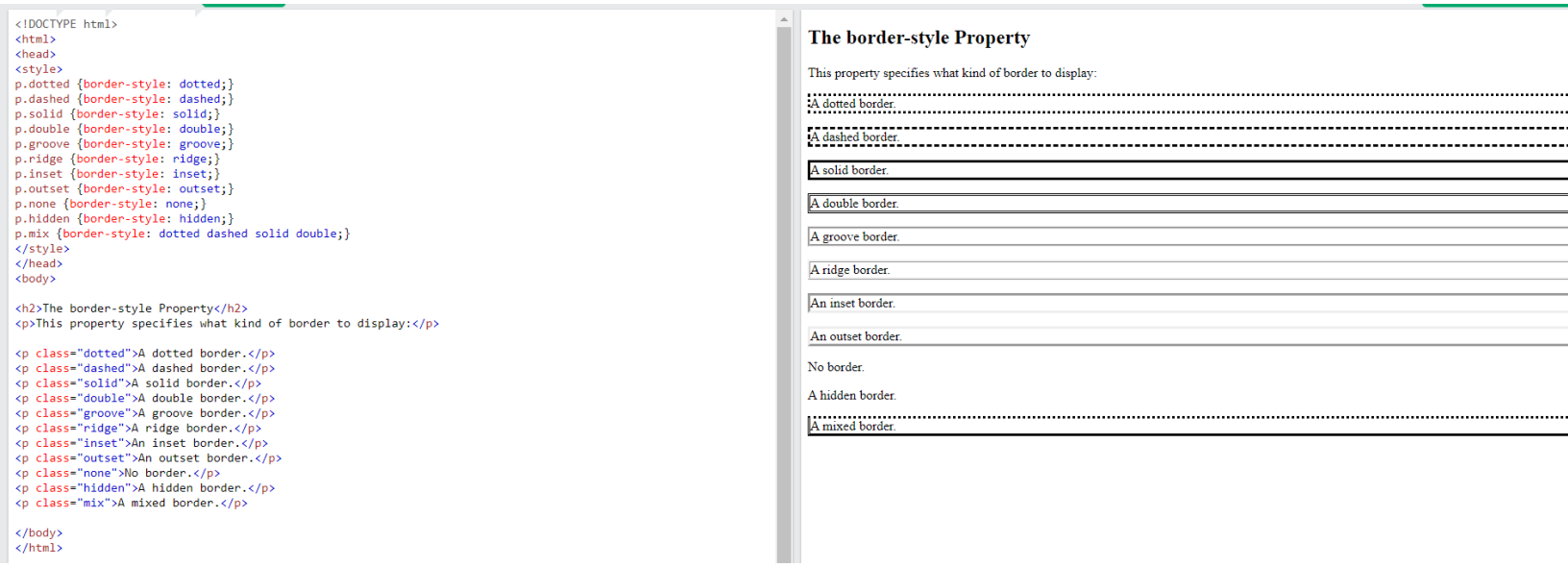
The `border-style` property specifies what kind of border to display.

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

The `border-style` property can have from one to four values (for the top border, right border, bottom border, and the left border).

```
<!DOCTYPE html>
<html>
<head>
<style>
p.dotted {border-style: dotted;}
p.dashed {border-style: dashed;}
p.solid {border-style: solid;}
p.double {border-style: double;}
p.groove {border-style: groove;}
p.ridge {border-style: ridge;}
p.inset {border-style: inset;}
p.outset {border-style: outset;}
p.none {border-style: none;}
p.hidden {border-style: hidden;}
p.mix {border-style: dotted dashed solid double;}
</style>
</head>
<body>

<h2>The border-style Property</h2>
<p>This property specifies what kind of border to display:</p>

<p class="dotted">A dotted border.</p>
<p class="dashed">A dashed border.</p>
<p class="solid">A solid border.</p>
<p class="double">A double border.</p>
<p class="groove">A groove border.</p>
<p class="ridge">A ridge border.</p>
<p class="inset">An inset border.</p>
<p class="outset">An outset border.</p>
<p class="none">No border.</p>
<p class="hidden">A hidden border.</p>
<p class="mix">A mixed border.</p>

</body>
</html>
```

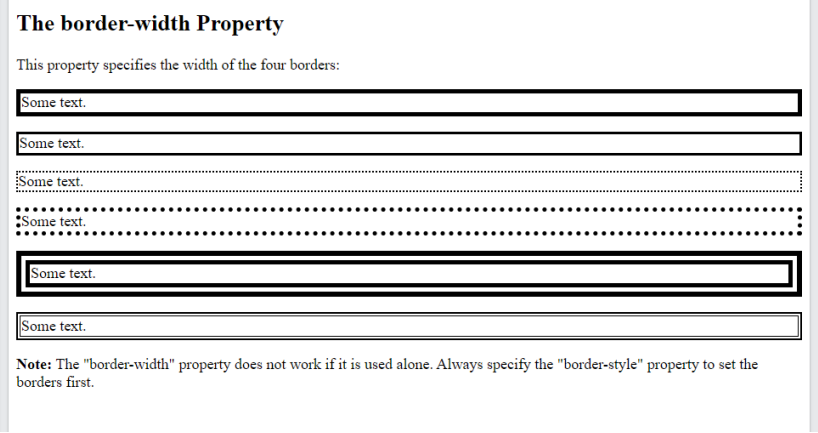


Border Width

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:

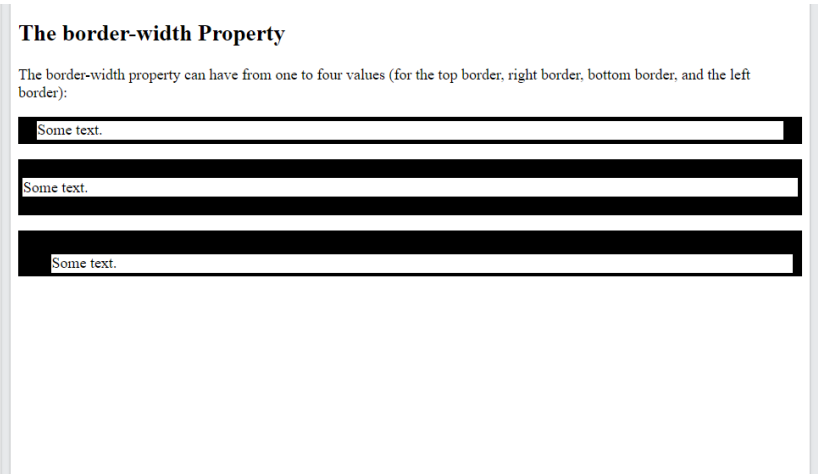
```
<!DOCTYPE html>
<html>
<head>
<style>
p.one { border-style: solid; border-width: 5px;}
p.two { border-style: solid; border-width: medium;}
p.three {border-style: dotted; border-width: 2px;}
p.four {border-style: dotted; border-width: thick;}
p.five {border-style: double; border-width: 15px;}
p.six {border-style: double; border-width: thick;}
</style>
</head>
<body>
<h2>The border-width Property</h2>
<p>This property specifies the width of the four borders:</p>
<p class="one">Some text.</p>
<p class="two">Some text.</p>
<p class="three">Some text.</p>
<p class="four">Some text.</p>
<p class="five">Some text.</p>
<p class="six">Some text.</p>
<p><b>Note:</b> The "border-width" property does not work if it is used alone.
Always specify the "border-style" property to set the borders first.</p>
</body>
</html>
```



Specific Side Widths

The `border-width` property can have from one to four values (for the top border, right border, bottom border, and the left border):

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  border-style: solid;
  border-width: 5px 20px; /* 5px top and bottom, 20px on the sides */
}
p.two {
  border-style: solid;
  border-width: 20px 5px; /* 20px top and bottom, 5px on the sides */
}
p.three {
  border-style: solid;
  border-width: 25px 10px 4px 35px; /* 25px top, 10px right, 4px bottom and 35px left */
}
</style>
</head>
<body>
<h2>The border-width Property</h2>
<p>The border-width property can have from one to four values (for the top border, right border,
bottom border, and the left border):</p>
<p class="one">Some text.</p>
<p class="two">Some text.</p>
<p class="three">Some text.</p>
</body>
</html>
```



Border Color

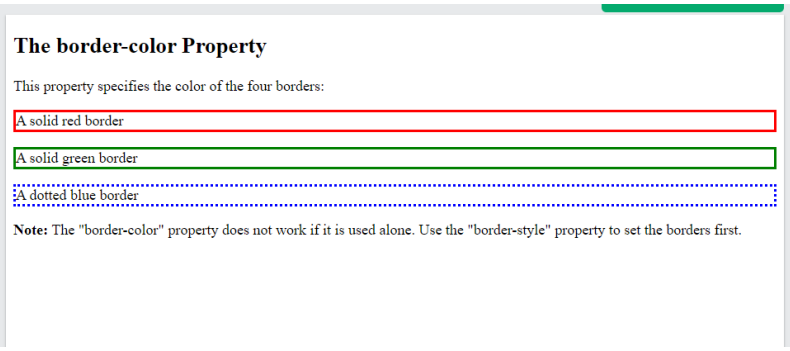
The `border-color` property is used to set the color of the four borders.

The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a HEX value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"
- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- transparent

Note: If `border-color` is not set, it inherits the color of the element.

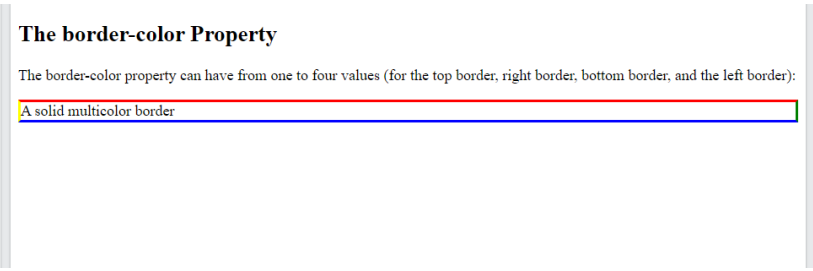
```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {border-style: solid; border-color: red;}
p.two {border-style: solid; border-color: green;}
p.three {border-style: dotted; border-color: blue;}
</style>
</head>
<body>
<h2>The border-color Property</h2>
<p>This property specifies the color of the four borders:</p>
<p class="one">A solid red border</p>
<p class="two">A solid green border</p>
<p class="three">A dotted blue border</p>
<p><b>Note:</b> The "border-color" property does not work if it is used alone. Use the "border-
style" property to set the borders first.</p>
</body>
</html>
```



Specific Side Colors

The `border-color` property can have from one to four values (for the top border, right border, bottom border, and the left border).

```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {border-style: solid;border-color: red green blue yellow;}
/* red top, green right, blue bottom and yellow left */
</style>
</head>
<body>
<h2>The border-color Property</h2>
<p>The border-color property can have from one to four values (for the top border, right border,
bottom border, and the left border):</p>
<p class="one">A solid multicolor border</p>
</body>
</html>
```



Border - Individual Sides

In CSS, there are also properties for specifying each of the borders (top, right, bottom, and left):

```
p {
  border-top-style: dotted;
  border-right-style: solid;
  border-bottom-style: dotted;
  border-left-style: solid;
}
```

Different Border Styles

The example above gives the same result as this:

```
p {
  border-style: dotted solid;
}
```

If the `border-style` property has four values:

- **border-style: dotted solid double dashed;**
  - top border is dotted
  - right border is solid
  - bottom border is double
  - left border is dashed

If the `border-style` property has three values:

- **border-style: dotted solid double;**
  - top border is dotted
  - right and left borders are solid
  - bottom border is double

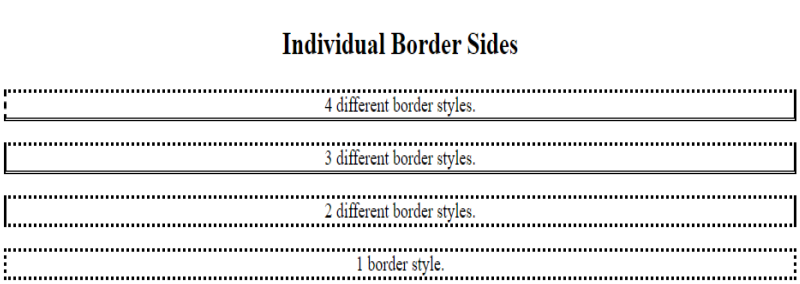
If the `border-style` property has two values:

- **border-style: dotted solid;**
  - top and bottom borders are dotted
  - right and left borders are solid

If the `border-style` property has one value:

- **border-style: dotted;**
  - all four borders are dotted

```
<!DOCTYPE html>
<html>
<head>
<style>
body {text-align: center;}
p.four {border-style: dotted solid double dashed;}/* Four values */
p.three {border-style: dotted solid double;}/* Three values */
p.two {border-style: dotted solid;} /* Two values */
p.one {border-style: dotted;} /* One value */
</style>
</head>
<body>
<h2>Individual Border Sides</h2>
<p class="four">4 different border styles.</p>
<p class="three">3 different border styles.</p>
<p class="two">2 different border styles.</p>
<p class="one">1 border style.</p>
</body>
</html>
```



Border - Shorthand Property

Like you saw in the previous page, there are many properties to consider when dealing with borders.

To shorten the code, it is also possible to specify all the individual border properties in one property.

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

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

```
p {
  border: 5px solid red;
}
```

Rounded Borders

The `border-radius` property is used to add rounded borders to an element:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  border: 2px solid red;
  padding: 5px;
}
p.round1 {
  border: 2px solid red;
  border-radius: 5px;
  padding: 5px;
}
p.round2 {
  border: 2px solid red;
  border-radius: 8px;
  padding: 5px;
}
p.round3 {
  border: 2px solid red;
  border-radius: 12px;
  padding: 5px;
}
</style>
</head>
<body>
<h2>The border-radius Property</h2>
<p>This property is used to add rounded borders to an element:</p>
<p class="normal">Normal border</p>
<p class="round1">Round border</p>
<p class="round2">Rounder border</p>
<p class="round3">Roundest border</p>
</body>
</html>
```

The border-radius Property

This property is used to add rounded borders to an element:

Normal border

Round border

Rounder border

Roundest border

Margins

The CSS `margin` properties are used to create space around elements, outside of any defined borders.

With CSS, you have full control over the margins. There are properties for setting the margin for each side of an element (top, right, bottom, and left).

Margin - Individual Sides

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 - the browser calculates the margin
- *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.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
  background-color: lightblue;
}
</style>
</head>
<body>
<h2>Using individual margin properties</h2>
<div>This div element has a top margin of 100px, a right margin of 150px, a bottom margin of 100px,
and a left margin of 80px.</div>
</body>
</html>
```

Using individual margin properties

This div element has a top margin of 100px, a right margin of 150px, a bottom margin of 100px, and a left margin of 80px.

Margin - Shorthand Property

If the `margin` property has four values:

- **margin: 25px 50px 75px 100px;**
  - top margin is 25px
  - right margin is 50px
  - bottom margin is 75px
  - left margin is 100px

Use the margin shorthand property with four values:

```
p {
  margin: 25px 50px 75px 100px;
}
```

If the `margin` property has three values:

- **margin: 25px 50px 75px;**
  - top margin is 25px
  - right and left margins are 50px
  - bottom margin is 75px

Use the margin shorthand property with three values:

```
p {
  margin: 25px 50px 75px;
}
```

If the `margin` property has two values:

- **margin: 25px 50px;**
  - top and bottom margins are 25px
  - right and left margins are 50px

Use the margin shorthand property with two values:

```
p {
  margin: 25px 50px;
}
```

If the `margin` property has one value:

- **margin: 25px;**
  - all four margins are 25px

Use the margin shorthand property with one value:

```
p {
  margin: 25px;
}
```

The auto MARGIN

You can set the margin property to `auto` to horizontally center the element within its container.

The element will then take up the specified width, and the remaining space will be split equally between the left and right margins.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 300px;
  margin: auto;
  border: 1px solid red;
}
</style>
</head>
<body>
<h2>Use of margin: auto</h2>
<p>You can set the margin property to auto to horizontally center the element within its container.
The element will then take up the specified width, and the remaining space will be split equally
between the left and right margins:</p>
<div>
This div will be horizontally centered because it has margin: auto;
</div>

</body>
</html>
```

Use of margin: auto

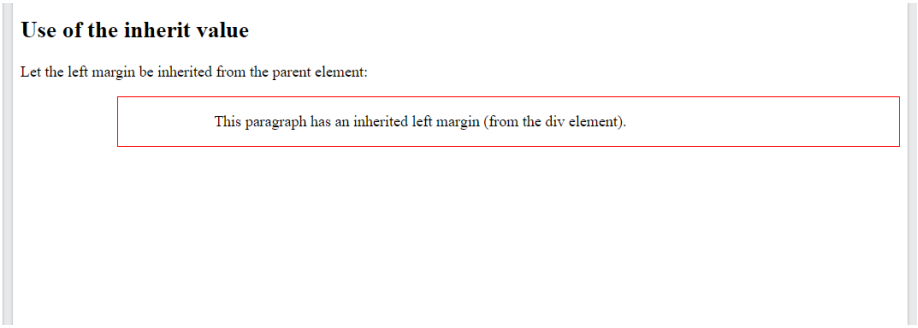
You can set the margin property to auto to horizontally center the element within its container. The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

This div will be horizontally centered because it has margin: auto;

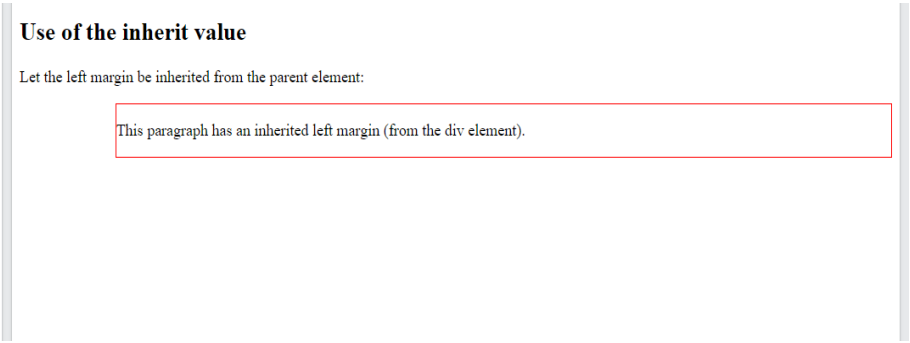
The inherit Value

This example lets the left margin of the <p class="ex1"> element be inherited from the parent element ( ):

```
<!DOCTYPE html><html><head>
<style>
div {
  border: 1px solid red;
  margin-left: 100px;
}

p.ex1 {
  margin-left: inherit;
}
</style>
</head>
<body>
<h2>Use of the inherit value</h2>
<p>Let the left margin be inherited from the parent element:</p>
<div>
<p class="ex1">This paragraph has an inherited left margin (from the div element).</p>
</div>
```




```
<!DOCTYPE html><html><head>
<style>
div {
  border: 1px solid red;
  margin-left: 100px;
}

p.ex1 {
  margin-right: inherit;
}
</style>
</head>
<body>
<h2>Use of the inherit value</h2>
<p>Let the left margin be inherited from the parent element:</p>
<div>
<p class="ex1">This paragraph has an inherited left margin (from the div element).</p>
</div>
```

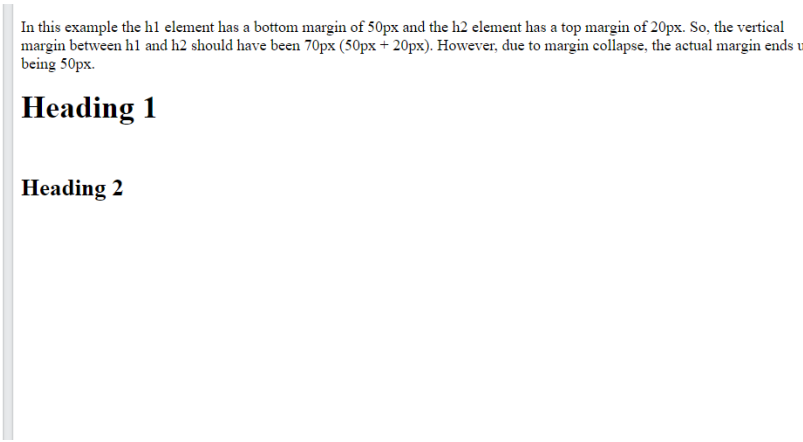


Margin Collapse

Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.

This does not happen on left and right margins! Only top and bottom margins!

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  margin: 0 0 50px 0;
}

h2 {
  margin: 20px 0 0 0;
}
</style>
</head>
<body>

<p>In this example the h1 element has a bottom margin of 50px and the h2 element has a top margin
of 20px. So, the vertical margin between h1 and h2 should have been 70px (50px + 20px). However,
due to margin collapse, the actual margin ends up being 50px.</p>

<h1>Heading 1</h1>
<h2>Heading 2</h2>

</body>
</html>
```



In the example above, the <h1> element has a bottom margin of 50px and the <h2> element has a top margin set to 20px.

Common sense would seem to suggest that the vertical margin between the <h1> and the <h2> would be a total of 70px (50px + 20px). But due to margin collapse, the actual margin ends up being 50px.

Padding

The CSS `padding` properties are used to generate space around an element's content, inside of any defined borders.

With CSS, you have full control over the padding. There are properties for setting the padding for each side of an element (top, right, bottom, and left).

Padding - Individual Sides

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

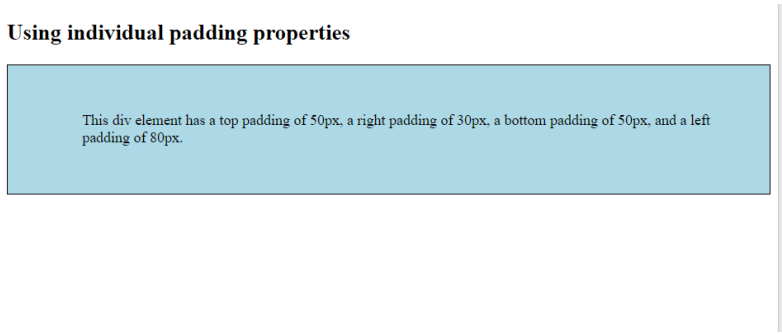
- `padding-top`
- `padding-right`
- `padding-bottom`
- `padding-left`

All the padding properties can have the following values:

- *length* - specifies a padding in px, pt, cm, etc.
- % - specifies a padding in % of the width of the containing element
- inherit - specifies that the padding should be inherited from the parent element

Note: Negative values are not allowed.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  background-color: lightblue;
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
</style>
</head>
<body>
<h2>Using individual padding properties</h2>
<div>This div element has a top padding of 50px, a right padding of 30px, a bottom padding of 50px,
and a left padding of 80px.</div>
```



Padding - Shorthand Property

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

The `padding` property is a shorthand property for the following individual padding properties:

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

So, here is how it works:

If the `padding` property has four values:

- **padding: 25px 50px 75px 100px;**
  - top padding is 25px
  - right padding is 50px
  - bottom padding is 75px
  - left padding is 100px

Use the padding shorthand property with four values:

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

If the `padding` property has three values:

- **padding: 25px 50px 75px;**
  - top padding is 25px
  - right and left paddings are 50px
  - bottom padding is 75px

Use the padding shorthand property with three values:

```
div {
  padding: 25px 50px 75px;
}
```

If the `padding` property has two values:

- **padding: 25px 50px;**
  - top and bottom paddings are 25px
  - right and left paddings are 50px

Use the padding shorthand property with two values:

```
div {
  padding: 25px 50px;
}
```

If the `padding` property has one value:

- **padding: 25px;**
  - all four paddings are 25px

Use the padding shorthand property with one value:

```
div {
  padding: 25px;
}
```

Padding and Element Width

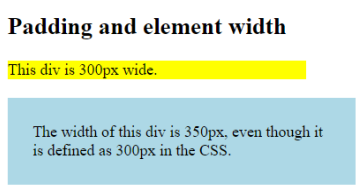
The CSS `width` property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element (the box model).

So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result.

Here, the element is given a width of 300px. However, the actual width of the element will be 350px (300px + 25px of left padding + 25px of right padding):

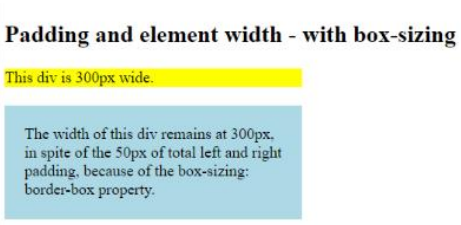
```
div {
  width: 300px;
  padding: 25px;
}
```

```
<head>
<style>
div.ex1 {
  width: 300px;
  background-color: yellow;}
div.ex2 {
  width: 300px;
  padding: 25px;
  background-color: lightblue;}
</style>
</head>
<body>
<h2>Padding and element width</h2>
<div class="ex1">This div is 300px wide.</div>
<br>
<div class="ex2">The width of this div is 350px, even though it is defined as 300px in the CSS.
</div>
```



To keep the width at 300px, no matter the amount of padding, you can use the `box-sizing` property. This causes the element to maintain its actual width; if you increase the padding, the available content space will decrease.

```
<head>
<style>
div.ex1 {
  width: 300px;
  background-color: yellow;}
div.ex2 {
  width: 300px;
  padding: 20px;
  box-sizing: border-box;
  background-color: lightblue;}
</style>
</head>
<body>
<h2>Padding and element width - with box-sizing</h2>
<div class="ex1">This div is 300px wide.</div>
<br>
<div class="ex2">The width of this div remains at 300px, in spite of the 50px of total left and
right padding, because of the box-sizing: border-box property.
</div>
```



Height, Width and Max-width

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

The CSS `max-width` property is used to set the maximum width of an element.

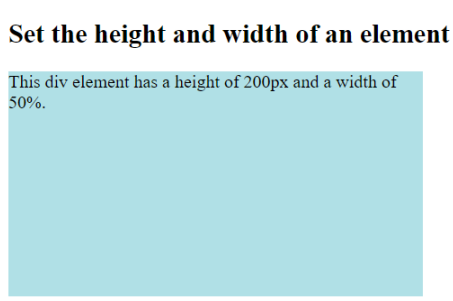
The height and width properties do not include padding, borders, or margins. It sets the height/width of the area inside the padding, border, and margin of the element.

CSS height and width Values

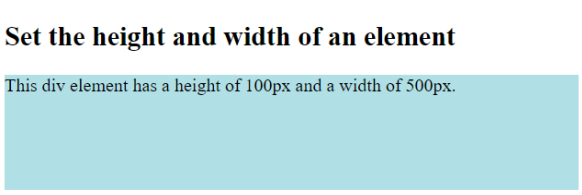
The `height` and `width` properties may have the following values:

- `auto` - This is default. The browser calculates the height and width
- `length` - Defines the height/width in px, cm, etc.
- `%` - Defines the height/width in percent of the containing block
- `initial` - Sets the height/width to its default value
- `inherit` - The height/width will be inherited from its parent value

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
</style>
</head>
<body>

<h2>Set the height and width of an element</h2>

<div>This div element has a height of 200px and a width of 50%.</div>

</body>
</html>
```




```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  height: 100px;
  width: 500px;
  background-color: powderblue;}
</style>
</head>
<body>
<h2>Set the height and width of an element</h2>
<div>This div element has a height of 100px and a width of 500px.</div>
</body>
</html>
```



Setting max-width

The `max-width` property is used to set the maximum width of an element.

The `max-width` can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

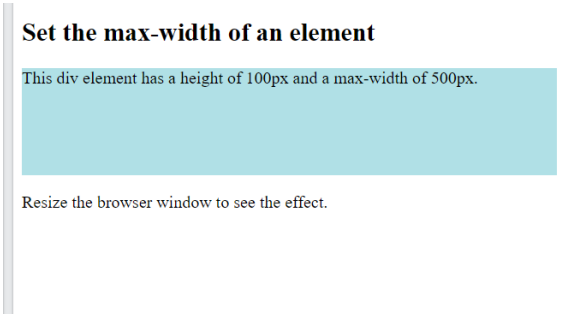
The problem with the `<div>` above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows.

Tip: Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

Note: If you for some reason use both the `width` property and the `max-width` property on the same element, and the value of the `width` property is larger than the `max-width` property; the `max-width` property will be used (and the `width` property will be ignored).

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  max-width: 500px;
  height: 100px;
  background-color: powderblue;}
</style>
</head>
<body>
<h2>Set the max-width of an element</h2>
<div>This div element has a height of 100px and a max-width of 500px.</div>
<p>Resize the browser window to see the effect.</p>
</body>
</html>
```

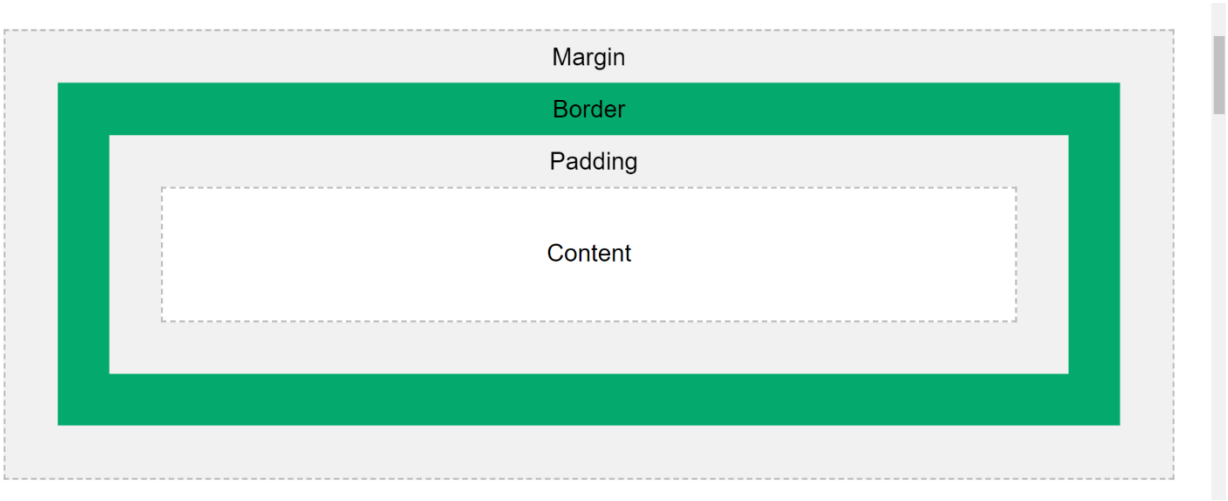


Box Model

In CSS, the term "box model" is used when talking about design and layout.

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

The image below illustrates the box model:



Explanation of the different parts:

- **Content** - The content of the box, where text and images appear
- **Padding** - Clears an area around the content. The padding is transparent
- **Border** - A border that goes around the padding and content
- **Margin** - Clears an area outside the border. The margin is transparent

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: lightgrey;
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
</style>
</head>
<body>
<h2>Demonstrating the Box Model</h2>
<p>The CSS box model is essentially a box that wraps around every HTML element. It
consists of: borders, padding, margins, and the actual content.</p>
<div>This text is the content of the box. We have added a 50px padding, 20px margin and a
15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in
voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat
cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.
</div>

</body>
</html>
```

Demonstrating the Box Model

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

This text is the content of the box. We have added a 50px padding, 20px margin and a 15px green border. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Width and Height of an Element

In order to set the width and height of an element correctly in all browsers, you need to know how the box model works.

Important: 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 full size of an element, you must also add padding, borders and margins.

This element will have a total width of 350px:

```
div {
  width: 320px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
```

Here is the calculation:

320px (width)
+ 20px (left + right padding)
+ 10px (left + right border)
+ 0px (left + right margin)
= 350px

The total width of an element should be calculated like this:

Total element width = width + left padding + right padding + left border + right border + left margin + right margin

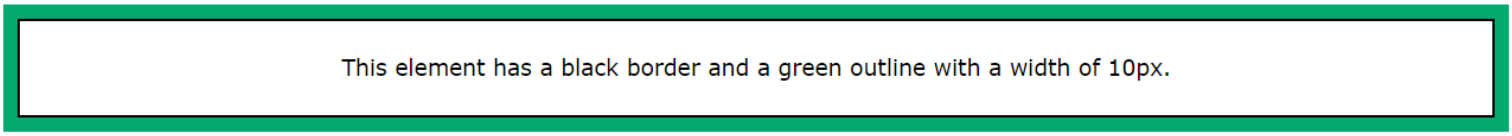
The total height of an element should be calculated like this:

Total element height = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

Outline

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

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out".



CSS has the following outline properties:

- `outline-style`
- `outline-color`
- `outline-width`
- `outline-offset`
- `outline`

CSS Outline Style

The `outline-style` property specifies the style of the outline, and can have one of the following values:

- `dotted` - Defines a dotted outline
- `dashed` - Defines a dashed outline
- `solid` - Defines a solid outline
- `double` - Defines a double outline
- `groove` - Defines a 3D grooved outline
- `ridge` - Defines a 3D ridged outline
- `inset` - Defines a 3D inset outline
- `outset` - Defines a 3D outset outline
- `none` - Defines no outline
- `hidden` - Defines a hidden outline

```
<!DOCTYPE html>
<html>
<head>
<style>
p {outline-color:red;}
p.dotted {outline-style: dotted;}
p.dashed {outline-style: dashed;}
p.solid {outline-style: solid;}
p.double {outline-style: double;}
p.groove {outline-style: groove;}
p.ridge {outline-style: ridge;}
p.inset {outline-style: inset;}
p.outset {outline-style: outset;}
</style>
</head>
<body>
<h2>The outline-style Property</h2>
<p class="dotted">A dotted outline</p>
<p class="dashed">A dashed outline</p>
<p class="solid">A solid outline</p>
<p class="double">A double outline</p>
<p class="groove">A groove outline. The effect depends on the outline-color value.</p>
<p class="ridge">A ridge outline. The effect depends on the outline-color value.</p>
<p class="inset">An inset outline. The effect depends on the outline-color value.</p>
<p class="outset">An outset outline. The effect depends on the outline-color value.</p>
</body>
</html>
```

The outline-style Property

A dotted outline

A dashed outline

A solid outline

A double outline

A groove outline. The effect depends on the outline-color value.

A ridge outline. The effect depends on the outline-color value.

An inset outline. The effect depends on the outline-color value.

An outset outline. The effect depends on the outline-color value.

CSS Outline Width

The `outline-width` property specifies the width of the outline, and can have one of the following values:

- thin (typically 1px)
- medium (typically 3px)
- thick (typically 5px)
- A specific size (in px, pt, cm, em, etc)

The following example shows some outlines with different widths:

A thin outline.

A medium outline.

A thick outline.

A 4px thick outline.

CSS Outline Color

The `outline-color` property is used to set the color of the outline.
The color can be set by:

- name - specify a color name, like "red"
- HEX - specify a hex value, like "#ff0000"
- RGB - specify a RGB value, like "rgb(255,0,0)"

- HSL - specify a HSL value, like "hsl(0, 100%, 50%)"
- invert - performs a color inversion (which ensures that the outline is visible, regardless of color background)

CSS Outline - Shorthand property

The `outline` property is a shorthand property for setting the following individual outline properties:

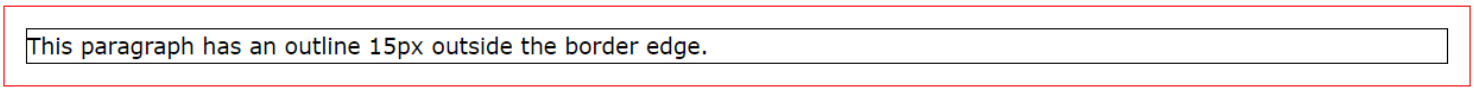
- `outline-width`
- `outline-style` (required)
- `outline-color`

```
p.ex1 {outline: dashed;}
p.ex2 {outline: dotted red;}
p.ex3 {outline: 5px solid yellow;}
p.ex4 {outline: thick ridge pink;}
```

CSS Outline Offset

The `outline-offset` property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.

The following example specifies an outline 15px outside the border edge:



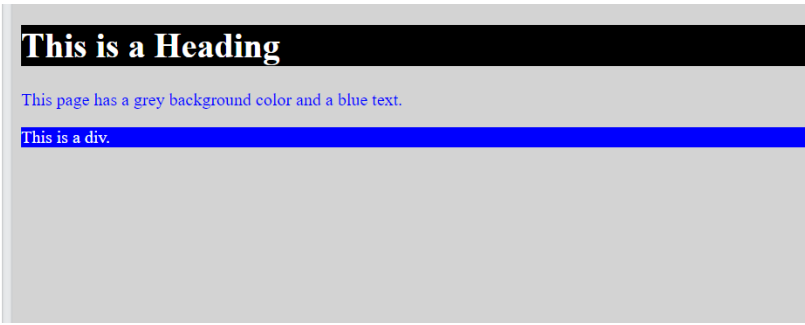
Example

```
p {
  margin: 30px;
  border: 1px solid black;
  outline: 1px solid red;
  outline-offset: 15px;
}
```

Text

TEXT COLOR

```
<!DOCTYPE html>
<html>
<head>
<style>
body { background-color: lightgrey; color: blue;}
h1 {background-color: black; color: white;}
div {background-color: blue; color: white;}
</style>
</head>
<body>
<h1>This is a Heading</h1>
<p>This page has a grey background color and a blue text.</p>
<div>This is a div.</div>
</body>
</html>
```



Text Alignment and Text Direction

- text-align
- text-align-last
- direction
- unicode-bidi
- vertical-align

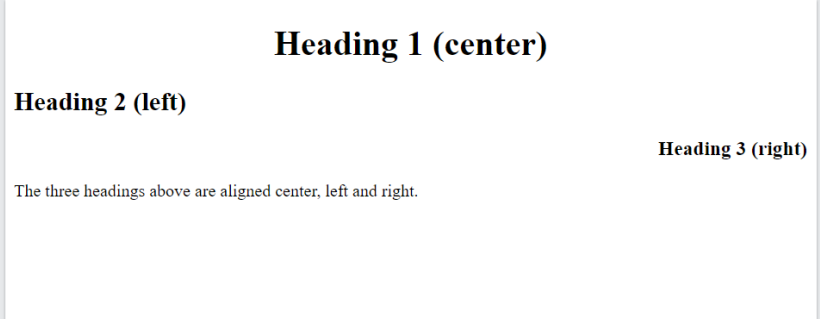
Text Alignment

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.

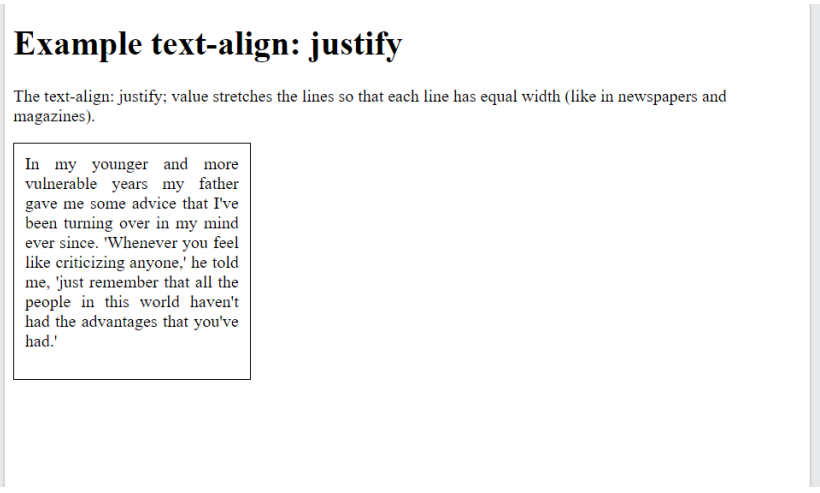
The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {text-align: center}
h2 {text-align: left;}
h3 {text-align: right;}
</style>
</head>
<body>
<h1>Heading 1 (center)</h1>
<h2>Heading 2 (left)</h2>
<h3>Heading 3 (right)</h3>
<p>The three headings above are aligned center, left and right.</p>
</body>
</html>
```



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):

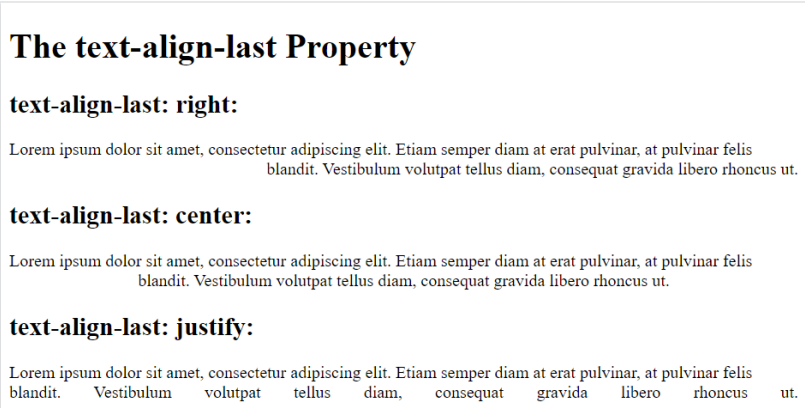
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 1px solid black;
  padding: 10px;
  width: 200px;
  height: 200px;
  text-align: justify;
}
</style>
</head>
<body>
<h1>Example text-align: justify</h1>
<p>The text-align: justify; value stretches the lines so that each line has equal width (like in newspapers and magazines).</p>
<div>
In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel like criticizing anyone,' he told me, 'just remember that all the people in this world haven't had the advantages that you've had.'
</div>
</body>
</html>
```



Text Align Last

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

```
p.c {
  text-align-last: justify;
}
</style>
</head>
<body>

<h1>The text-align-last Property</h1>

<h2>text-align-last: right:</h2>
<p class="a">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.</p>

<h2>text-align-last: center:</h2>
<p class="b">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.</p>

<h2>text-align-last: justify:</h2>
<p class="c">Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit. Vestibulum volutpat tellus diam, consequat gravida libero rhoncus ut.</p>

</body>
</html>
```



Text Direction

The `direction` and `unicode-bidi` properties can be used to change the text direction of an element:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
  direction: rtl;
  unicode-bidi: bidi-override;
}
</style>
</head>
<body>
<p>This is the default text direction.</p>
<p class="ex1">This is right-to-left text direction.</p>
</body>
</html>
```

Vertical Alignment

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

```
<!DOCTYPE html>
<html>
<head>
<style>
img.a {vertical-align: baseline;}
img.b {vertical-align: text-top;}
img.c {vertical-align: text-bottom;}
img.d {vertical-align: sub;}
img.e {vertical-align: super;}
</style>
</head>
<body>
<h1>The vertical-align Property</h1>
<h2>vertical-align: baseline (default):</h2>
<p>An <img class="a" src="sqpurple.gif" width="9" height="9"> image with a default
alignment.</p>
<h2>vertical-align: text-top:</h2>
<p>An <img class="b" src="sqpurple.gif" width="9" height="9"> image with a text-top
alignment.</p>
<h2>vertical-align: text-bottom:</h2>
<p>An <img class="c" src="sqpurple.gif" width="9" height="9"> image with a text-bottom
alignment.</p>
<h2>vertical-align: sub:</h2>
<p>An <img class="d" src="sqpurple.gif" width="9" height="9"> image with a sub
alignment.</p>
<h2>vertical-align: sup:</h2>
<p>An <img class="e" src="sqpurple.gif" width="9" height="9"> image with a super
alignment.</p>
</body>
</html>
```

The vertical-align Property

vertical-align: baseline (default):

An image with a default alignment.

vertical-align: text-top:

An image with a text-top alignment.

vertical-align: text-bottom:

An image with a text-bottom alignment.

vertical-align: sub:

An image with a sub alignment.

vertical-align: sup:

An image with a super alignment.

Text Decoration

- `text-decoration-line`
- `text-decoration-color`
- `text-decoration-style`
- `text-decoration-thickness`
- `text-decoration`

Decoration Line to Text

The `text-decoration-line` property is used to add a decoration line to text.

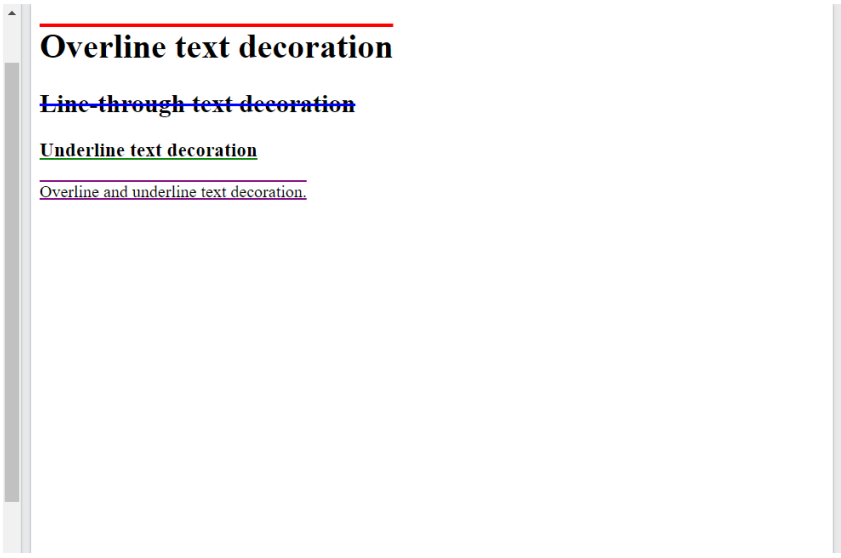
Tip: You can combine more than one value, like overline and underline to display lines both over and under a text.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration: overline;
}

h2 {
  text-decoration: line-through;
}

h3 {
  text-decoration: underline;
}

p.ex {
  text-decoration: overline underline;
}
</style>
</head>
<body>
<h1>Overline text decoration</h1>
<h2>Line-through text decoration</h2>
<h3>Underline text decoration</h3>
<p class="ex">Overline and underline text decoration.</p>
<p><strong>Note:</strong> It is not recommended to underline text that is not a link,
as this often confuses
the reader.</p>
</body>
```

Specify a Color for the Decoration Line

The `text-decoration-color` property is used to set the color of the decoration line.

```
<style>
h1 {
  text-decoration-line: overline;
  text-decoration-color: red;
}

h2 {
  text-decoration-line: line-through;
  text-decoration-color: blue;
}

h3 {
  text-decoration-line: underline;
  text-decoration-color: green;
}

p {
  text-decoration-line: overline underline;
  text-decoration-color: purple;
}
</style>
</head>
<body>
<h1>Overline text decoration</h1><h2>Line-through text decoration</h2>
<h3>Underline text decoration</h3>
<p>Overline and underline text decoration.</p>

</body>
</html>
```



Specify a Style for the Decoration Line

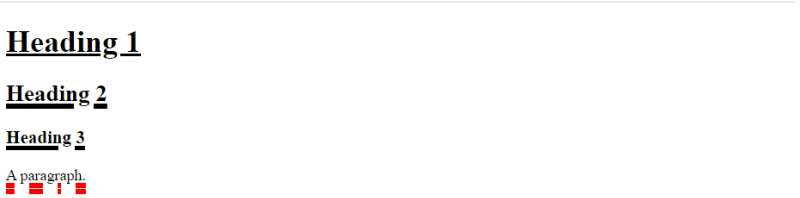
The `text-decoration-style` property is used to set the style of the decoration line.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration-line: underline;
  text-decoration-style: solid; /* this is default */}
h2 {
  text-decoration-line: underline;
  text-decoration-style: double;}
h3 {
  text-decoration-line: underline;
  text-decoration-style: dotted;  }
p.ex1 {
  text-decoration-line: underline;
  text-decoration-style: dashed;  }
p.ex2 {
  text-decoration-line: underline;
  text-decoration-style: wavy;  }
p.ex3 {
  text-decoration-line: underline;
  text-decoration-color: red;
  text-decoration-style: wavy;  }
</style>
</head>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<p class="ex1">A paragraph.</p>
<p class="ex2">Another paragraph.</p>
<p class="ex3">Another paragraph.</p>
</body>
</html>
```

Specify the Thickness for the Decoration Line

The `text-decoration-thickness` property is used to set the thickness of the decoration line.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  text-decoration-line: underline;
  text-decoration-thickness: auto;  /* this is default */
}
h2 {
  text-decoration-line: underline;
  text-decoration-thickness: 5px;
}
h3 {
  text-decoration-line: underline;
  text-decoration-thickness: 25%;
}
p {
  text-decoration-line: underline;
  text-decoration-color: red;
  text-decoration-style: double;
  text-decoration-thickness: 5px;
}
</style>
</head>
<body>
<h1>Heading 1</h1>
<h2>Heading 2</h2>
<h3>Heading 3</h3>
<p>A paragraph.</p>

</body>
</html>
```



The Shorthand Property

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)

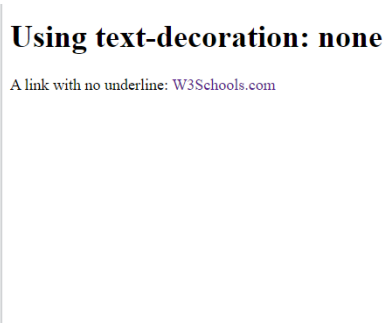
```
p {
  text-decoration: underline red double 5px;
}
```

A Small Tip

All links in HTML are underlined by default. Sometimes you see that links are styled with no underline. The `text-decoration: none;` is used to remove the underline from links, like this:

```
a {
  text-decoration: none;
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
a {
  text-decoration: none;
}
</style>
</head>
<body>

<h1>Using text-decoration: none</h1>

<p>A link with no underline: <a href="https://www.w3schools.com">W3Schools.com</a></p>

</body>
</html>
```

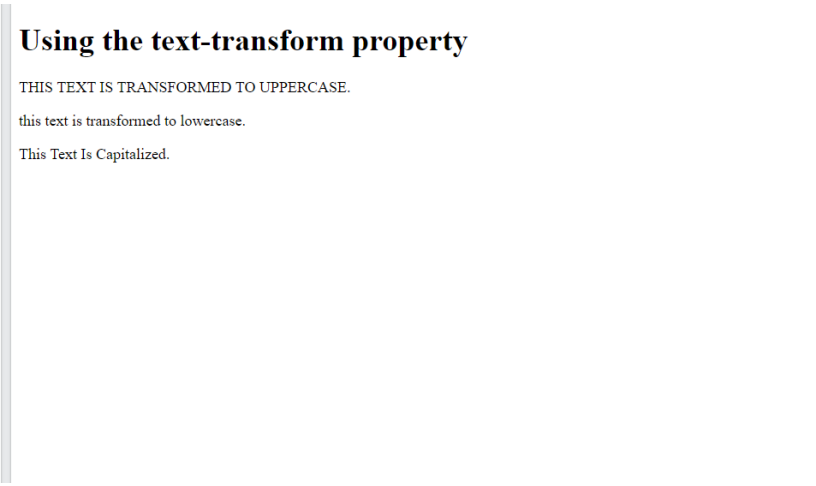


Text Transformation

The `text-transform` property is used to specify uppercase and lowercase letters in a text.

It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
</style>
</head>
<body>

<h1>Using the text-transform property</h1>

<p class="uppercase">This text is transformed to uppercase.</p>
<p class="lowercase">This text is transformed to lowercase.</p>
<p class="capitalize">This text is capitalized.</p>

</body>
</html>
```



Text Spacing

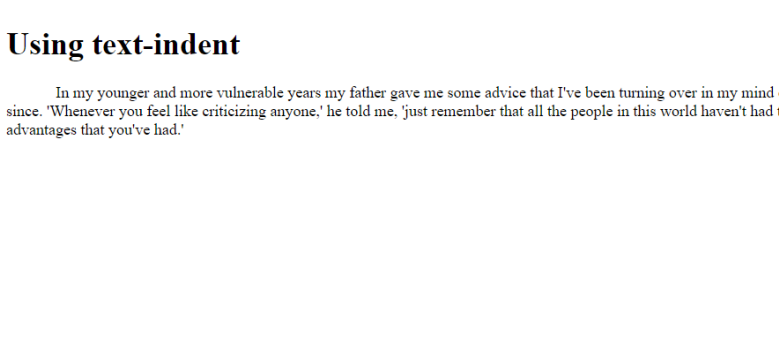
Text Indentation, Letter Spacing, Line Height, Word Spacing, and White Space

- `text-indent`
- `letter-spacing`
- `line-height`
- `word-spacing`
- `white-space`

Text Indentation

The `text-indent` property is used to specify the indentation of the first line of a text:

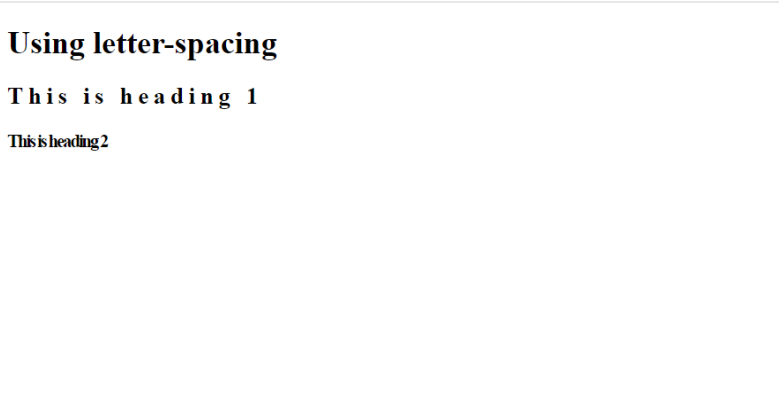
```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  text-indent: 50px;
}
</style>
</head>
<body>

<h1>Using text-indent</h1>

<p>In my younger and more vulnerable years my father gave me some advice that I've been turning over in my mind ever since. 'Whenever you feel like criticizing anyone,' he told me, 'just remember that all the people in this world haven't had the advantages that you've had.'</p>

</body>
</html>
```



Letter Spacing

The `letter-spacing` property is used to specify the space between the characters in a text.

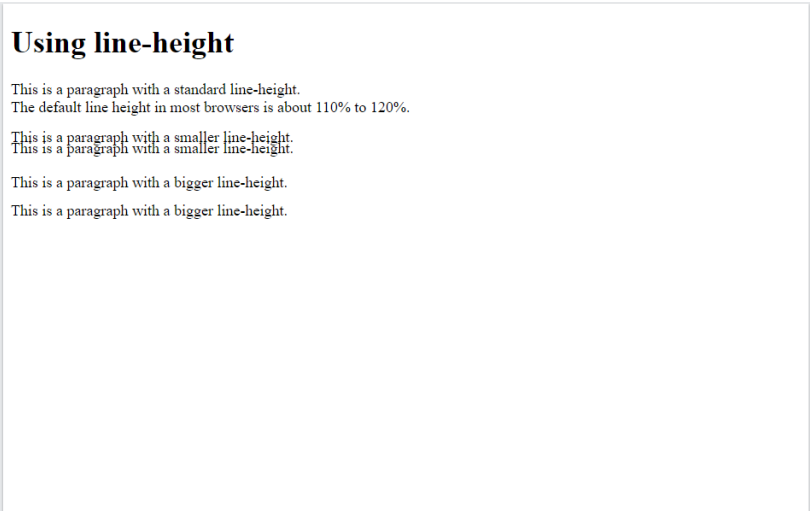
```
<!DOCTYPE html>
<html>
<head>
<style>
h2 {
  letter-spacing: 5px;
}

h3 {
  letter-spacing: -2px;
}
</style>
</head>
<body>

<h1>Using letter-spacing</h1>

<h2>This is heading 1</h2>
<h3>This is heading 2</h3>

</body>
</html>
```



Line Height

The `line-height` property is used to specify the space between lines:

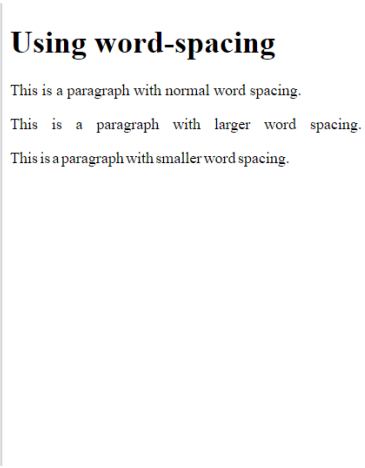
```
<!DOCTYPE html>
<html>
<head>
<style>
p.small {
  line-height: 0.7;
}

p.big {
  line-height: 1.8;
}
</style>
</head>
<body>
<h1>Using line-height</h1>
<p>
This is a paragraph with a standard line-height.<br>
The default line height in most browsers is about 110% to 120%.<br>
</p>
<p class="small">
This is a paragraph with a smaller line-height.<br>
This is a paragraph with a smaller line-height.<br>
</p>
<p class="big">
This is a paragraph with a bigger line-height.<br>
This is a paragraph with a bigger line-height.<br>
</p>
</body>
</html>
```



Word Spacing

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

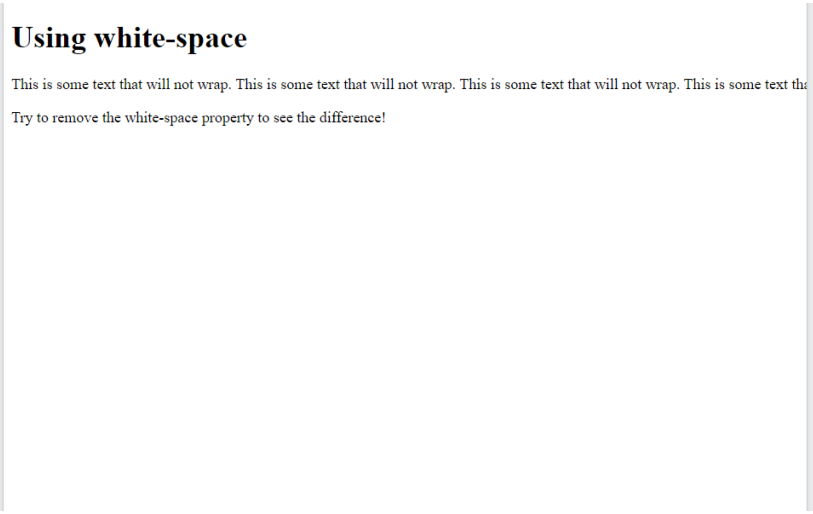
```
<!DOCTYPE html>
<html>
<head>
<style>
p.one {
  word-spacing: 10px;
}

p.two {
  word-spacing: -2px;
}
</style>
</head>
<body>

<h1>Using word-spacing</h1>

<p>This is a paragraph with normal word spacing.</p>

<p class="one">This is a paragraph with larger word spacing.</p>

<p class="two">This is a paragraph with smaller word spacing.</p>

</body>
</html>
```



White Space

The `white-space` property specifies how white-space inside an element is handled.

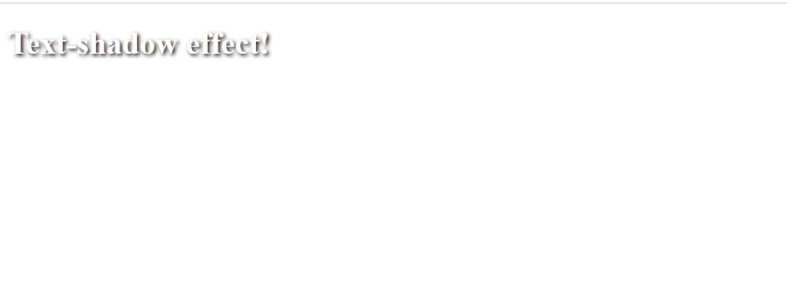
```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  white-space: nowrap;
}
</style>
</head>
<body>

<h1>Using white-space</h1>

<p>
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
This is some text that will not wrap.
</p>

<p>Try to remove the white-space property to see the difference!</p>

</body>
</html>
```



Text Shadow

The `text-shadow` property adds shadow to text.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: white;
  text-shadow: 2px 2px 4px #000000;
}
</style>
</head>
<body>

<h1>Text-shadow effect!</h1>

</body>
</html>
```



Fonts

Font Selection is Important

Choosing the right font has a huge impact on how the readers experience a website.

The right font can create a strong identity for your brand.

Using a font that is easy to read is important. The font adds value to your text. It is also important to choose the correct color and text size for the font.

Generic 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.

All the different font names belong to one of the generic font families.

Difference Between Serif and Sans-serif Fonts

Generic Font Family	Examples of Font Names
Serif	Times New Roman Georgia Garamond
Sans-serif	Arial Verdana Helvetica
Monospace	Courier New Lucida Console Monaco
Cursive	Brush Script MT Lucida Handwriting
Fantasy	COPPERPLATE Papyrus

The CSS font-family Property

In CSS, we use the `font-family` property to specify the font of a text.

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

```
<!DOCTYPE html>
<html>
<head>
<style>
.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
</style>
</head>
<body>

<h1>CSS font-family</h1>
<p class="p1">This is a paragraph, shown in the Times New Roman font.</p>
<p class="p2">This is a paragraph, shown in the Arial font.</p>
<p class="p3">This is a paragraph, shown in the Lucida Console font.</p>

</body>
</html>
```

CSS font-family

This is a paragraph, shown in the Times New Roman font.

This is a paragraph, shown in the Arial font.

This is a paragraph, shown in the Lucida Console font.

Web Safe Fonts

What are Web Safe Fonts?

Web safe fonts are fonts that are universally installed across all browsers and devices.

Fallback Fonts

However, there are no 100% completely web safe fonts. There is always a chance that a font is not found or is not installed properly.

Therefore, it is very important to always use fallback fonts.

This means that you should add a list of similar "backup fonts" in the `font-family` property. If the first font does not work, the browser will try the next one, and the next one, and so on. Always end the list with a generic font family name.

Here, there are three font types: Tahoma, Verdana, and sans-serif. The second and third fonts are backups, in case the first one is not found.

```
p {
font-family: Tahoma, Verdana, sans-serif;
}
```

Best Web Safe Fonts for HTML and CSS

The following list are the best web safe fonts for HTML and CSS:

- Arial (sans-serif)
- Verdana (sans-serif)
- Tahoma (sans-serif)
- Trebuchet MS (sans-serif)
- Times New Roman (serif)
- Georgia (serif)
- Garamond (serif)
- Courier New (monospace)
- Brush Script MT (cursive)

Note: Before you publish your website, always check how your fonts appear on different browsers and devices, and always use fallback fonts!

Font Style

The `font-style` property is mostly used to specify italic text.

This property has three values:

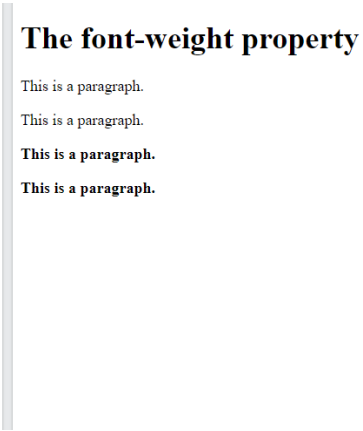
- normal - The text is shown normally
- italic - The text is shown in italics
- oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

```
<style>
p.normal {
  font-style: normal;
}

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
</style>
</head>
<body>
<h1>The font-style property</h1>
<p class="normal">This is a paragraph in normal style.</p>
<p class="italic">This is a paragraph in italic style.</p>
<p class="oblique">This is a paragraph in oblique style.</p>
```

Font Weight

The `font-weight` property specifies the weight of a font:

```
<style>
p.normal {
  font-weight: normal;
}
p.light {
  font-weight: lighter;
}
p.thick {
  font-weight: bold;
}
p.thicker {
  font-weight: 900;
}
</style>
</head>
<body>
<h1>The font-weight property</h1>
<p class="normal">This is a paragraph.</p>
<p class="light">This is a paragraph.</p>
<p class="thick">This is a paragraph.</p>
<p class="thicker">This is a paragraph.</p>

</body>
</html>
```

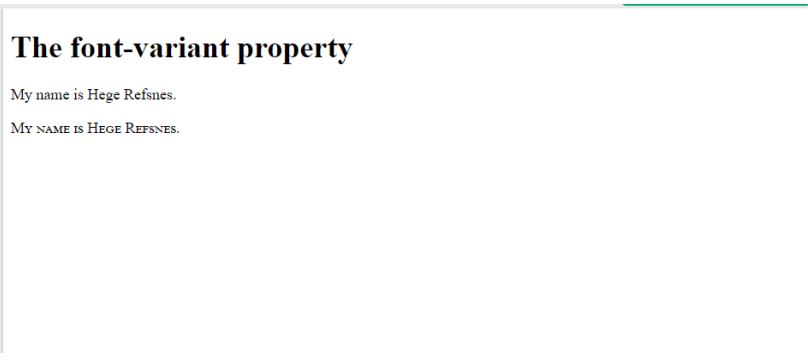


Font Variant

The `font-variant` property specifies whether or not a text should be displayed in a small-caps font.

In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appears in a smaller font size than the original uppercase letters in the text.

```
<!DOCTYPE html>
<html>
<head>
<style>
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
</style>
</head>
<body>
<h1>The font-variant property</h1>
<p class="normal">My name is Hege Refsnes.</p>
<p class="small">My name is Hege Refsnes.</p>

</body>
</html>
```



Font Size

The `font-size` property sets the size of the text.

Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs.

Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute, or relative size.

Absolute size:

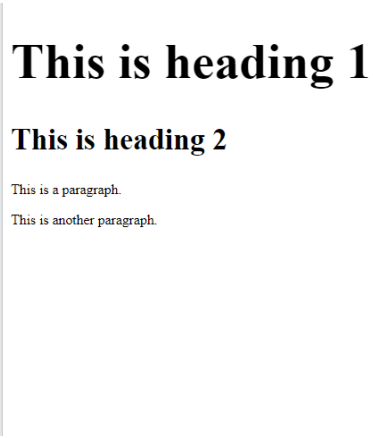
- Sets the text to a specified size
- Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
- Absolute size is useful when the physical size of the output is known

Relative size:

- Sets the size relative to surrounding elements
- Allows a user to change the text size in browsers

Note: If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 50px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
</style>
</head>
<body>
<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>This is another paragraph.</p>
</body>
</html>
```



Set Font Size With Em

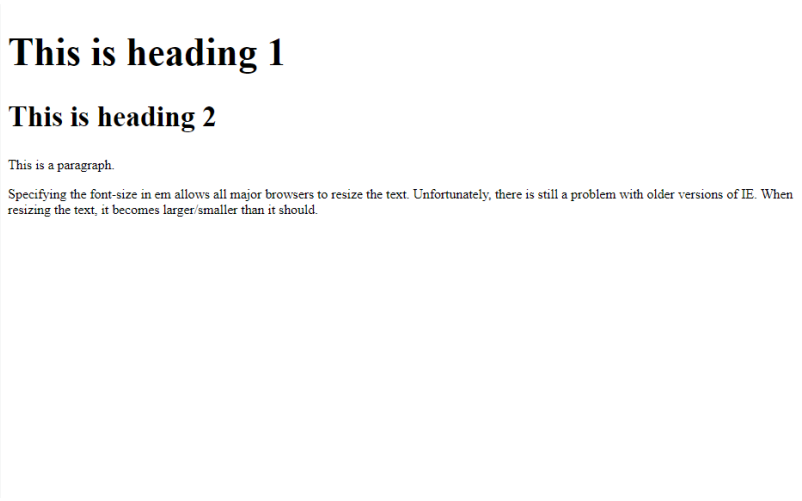
To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

In the example above, the text size in em is the same as the previous example in pixels. However, with the em size, it is possible to adjust the text size in all browsers.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  font-size: 2.5em; /* 40px/16=2.5em */
}

h2 {
  font-size: 1.875em; /* 30px/16=1.875em */
 }

p {
  font-size: 0.875em; /* 14px/16=0.875em */
}
</style>
</head>
<body>

<h1>This is heading 1</h1>
<h2>This is heading 2</h2>
<p>This is a paragraph.</p>
<p>Specifying the font-size in em allows all major browsers to resize the text.
Unfortunately, there is still a problem with older versions of IE. When resizing the text, it
becomes larger/smaller than it should.</p>

</body>
</html>
```



Google Fonts

Google Fonts

If you do not want to use any of the standard fonts in HTML, you can use Google Fonts.

Google Fonts are free to use, and have more than 1000 fonts to choose from.

How To Use Google Fonts

Just add a special style sheet link in the <head> section and then refer to the font in the CSS.

Here, we want to use a font named "Sofia" from Google Fonts:

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

<h1>Sofia Font</h1>
<p>Lorem ipsum dolor sit amet.</p>
<p>123456790</p>

</body>
</html>
```

Sofia Font

Lorem ipsum dolor sit amet.

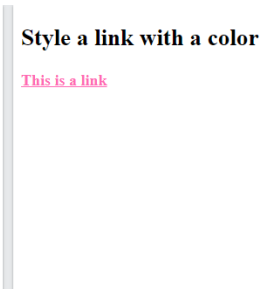
123456790

CSS Links

Styling Links

Links can be styled with any CSS property (e.g. `color`, `font-family`, `background`, etc.).

```
<!DOCTYPE html>
<html>
<head>
<style>
a {
  color: hotpink;
}
</style>
</head>
<body>
<h2>Style a link with a color</h2>
<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>

</body>
</html>
```



In addition, links can be styled differently depending on what **state** they are in.

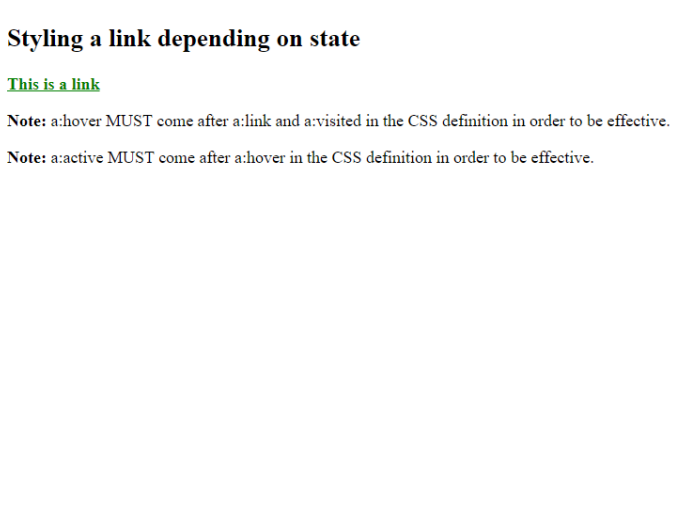
The four links states are:

- `a:link` - a normal, unvisited link
- `a:visited` - a link the user has visited
- `a:hover` - a link when the user mouses over it
- `a:active` - a link the moment it is clicked

When setting the style for several link states, there are some order rules:

- a:hover MUST come after a:link and a:visited
- a:active MUST come after a:hover

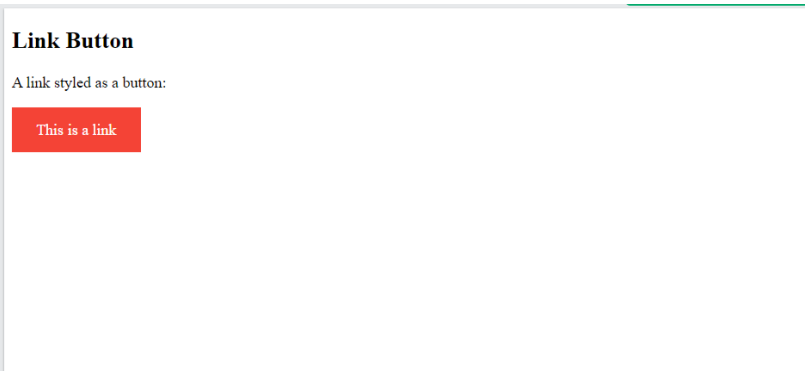
```
<style>
a:link {
  color: red;     /* unvisited link */
}

a:visited {
  color: green;  /* visited link */
}

a:hover {
  color: hotpink;  /* mouse over link */
}

a:active {
  color: blue;   /* selected link */
}
</style>
</head>
<body>
<h2>Styling a link depending on state</h2>

<p><b><a href="default.asp" target="_blank">This is a link</a></b></p>
<p><b>Note:</b> a:hover MUST come after a:link and a:visited in the CSS definition in order to be
effective.</p>
<p><b>Note:</b> a:active MUST come after a:hover in the CSS definition in order to be effective.
</p>
```



Link Buttons

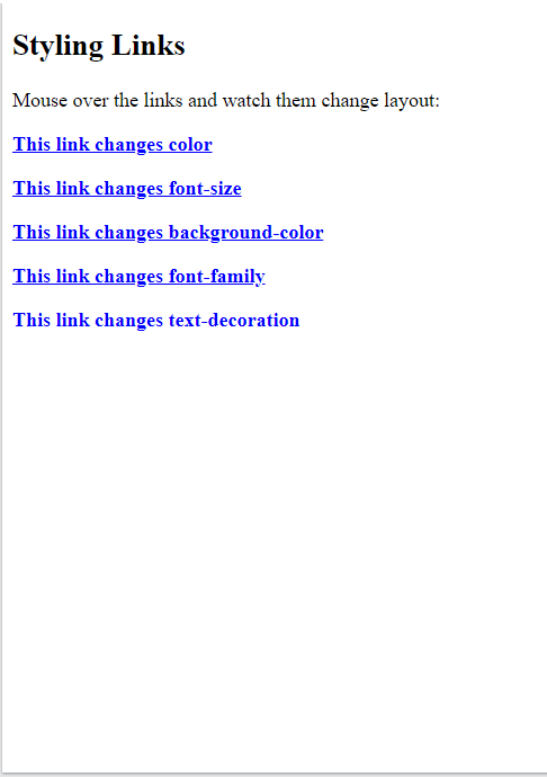
This example demonstrates a more advanced example where we combine several CSS properties to display links as boxes/buttons:

```
<!DOCTYPE html>
<html>
<head>
<style>
a:link, a:visited {
  background-color: #f44336;
  color: white;
  padding: 14px 25px;
  text-align: center;
  text-decoration: none;
  display: inline-block;}
a:hover, a:active {background-color: red;}
</style>
</head>
<body>
<h2>Link Button</h2>
<p>A link styled as a button:</p>
<a href="default.asp" target="_blank">This is a link</a>
</body>
</html>
```



This example demonstrates how to add other styles to hyperlinks:

```
<head>
<style>
a.one:link {color:#ff0000;}
a.one:visited {color:#0000ff;}
a.one:hover {color:#ffcc00;}

a.two:link {color:#ff0000;}
a.two:visited {color:#0000ff;}
a.two:hover {font-size:150%;}

a.three:link {color:#ff0000;}
a.three:visited {color:#0000ff;}
a.three:hover {background:#66ff66;}

a.four:link {color:#ff0000;}
a.four:visited {color:#0000ff;}
a.four:hover {font-family:monospace;}

a.five:link {color:#ff0000;text-decoration:none;}
a.five:visited {color:#0000ff;text-decoration:none;}
a.five:hover {text-decoration:underline;}
</style>
</head>
<body>
<h2>Styling Links</h2>
<p>Mouse over the links and watch them change layout:</p>
<p><b><a class="one" href="default.asp" target="_blank">This link changes color</a></b></p>
<p><b><a class="two" href="default.asp" target="_blank">This link changes font-size</a></b></p>
<p><b><a class="three" href="default.asp" target="_blank">This link changes background-color</a>
</b></p>
<p><b><a class="four" href="default.asp" target="_blank">This link changes font-family</a></b>
</p>
<p><b><a class="five" href="default.asp" target="_blank">This link changes text-decoration</a>
</b></p>
```



CURSOR

This example demonstrates the different types of cursors (can be useful for links):

```
<span style="cursor: auto">auto</span><br>
<span style="cursor: crosshair">crosshair</span><br>
<span style="cursor: default">default</span><br>
<span style="cursor: e-resize">e-resize</span><br>
<span style="cursor: help">help</span><br>
<span style="cursor: move">move</span><br>
<span style="cursor: n-resize">n-resize</span><br>
<span style="cursor: ne-resize">ne-resize</span><br>
<span style="cursor: nw-resize">nw-resize</span><br>
<span style="cursor: pointer">pointer</span><br>
<span style="cursor: progress">progress</span><br>
<span style="cursor: s-resize">s-resize</span><br>
<span style="cursor: se-resize">se-resize</span><br>
<span style="cursor: sw-resize">sw-resize</span><br>
<span style="cursor: text">text</span><br>
<span style="cursor: w-resize">w-resize</span><br>
<span style="cursor: wait">wait</span>
```

List

Unordered Lists:

- Coffee
- Tea
- Coca Cola

- Coffee
- Tea
- Coca Cola

Ordered Lists:

1. Coffee
2. Tea
3. Coca Cola

I. Coffee
II. Tea
III. Coca Cola

HTML Lists and CSS List Properties

In HTML, there are two main types of lists:

- unordered lists (<ul>) - the list items are marked with bullets
- ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

- Set different list item markers for ordered lists
- Set different list item markers for unordered lists
- Set an image as the list item marker
- Add background colors to lists and list items

list-style-type

The list-style-type property specifies the type of list item marker.

The following example shows some of the available list item markers:

```
<style>
ul.a {list-style-type: circle;}
ul.b {list-style-type: square;}
ol.c {list-style-type: upper-roman;}
ol.d {  list-style-type: lower-alpha;}
</style>
</head>
<body>
<h2>The list-style-type Property</h2>
<p>Example of unordered lists:</p>
<ul class="a">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
<ul class="b">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
<p>Example of ordered lists:</p>
<ol class="c">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
<ol class="d">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
</body>
</html>
```

The list-style-type Property

Example of unordered lists:

- Coffee
- Tea
- Coca Cola

- Coffee
- Tea
- Coca Cola

Example of ordered lists:

I. Coffee
II. Tea
III. Coca Cola

a. Coffee
b. Tea
c. Coca Cola

list-style-image

The `list-style-image` property specifies an image as the list item marker:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul {
  list-style-image: url('sqpurple.gif');}
</style>
</head>
<body>
<h2>The list-style-image Property</h2>
<p>The list-style-image property specifies an image as the list item marker:</p>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
</body>
</html>
```

The list-style-image Property

The list-style-image property specifies an image as the list item marker:

- Coffee
- Tea
- Coca Cola

list-style-position

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

"list-style-position: outside;" means that the bullet points will be outside the list item. The start of each line of a list item will be aligned vertically. This is default:

- Coffee - A brewed drink prepared from roasted coffee beans...
- Tea
- Coca-cola

"list-style-position: inside;" means that the bullet points will be inside the list item. As it is part of the list item, it will be part of the text and push the text at the start:

- Coffee - A brewed drink prepared from roasted coffee beans...
- Tea
- Coca-cola

```
      ul.a {
  list-style-position: outside;
}

ul.b {
  list-style-position: inside;
}
```

Remove Default Settings

The `list-style-type:none` property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add `margin:0` and `padding:0` to <ul> or <ol>:

```
<!DOCTYPE html>
<html>
<head>
<style>
ul.demo {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
</style>
</head>
<body>

<p>Default list:</p>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

<p>Remove bullets, margin and padding from list:</p>
<ul class="demo">
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>

</body>
</html>
```

Default list:

- Coffee
- Tea
- Coca Cola

Remove bullets, margin and padding from list:

Coffee
Tea
Coca Cola

Styling List With Colors

We can also style lists with colors, to make them look a little more interesting.

Anything added to the <ol> or <ul> tag, affects the entire list, while properties added to the <li> tag will affect the individual list items:

```
<head>
<style>
ol {
  background: #ff9999;
  padding: 20px;}
ul {
  background: #3399ff;
  padding: 20px;}
ol li {
  background: #ffe5e5;
  color: darkred;
  padding: 5px;
  margin-left: 35px;}
ul li {
  background: #cce5ff;
  color: darkblue;
  margin: 5px;}
</style>
</head>
<body>
<h1>Styling Lists With Colors</h1>
<ol>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ol>
<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Coca Cola</li>
</ul>
```

Styling Lists With Colors

1. Coffee
2. Tea
3. Coca Cola

- Coffee
- Tea
- Coca Cola

Tables

Table Borders

To specify table borders in CSS, use the `border` property.

The example below specifies a solid border for <table>, <th>, and <td> elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid;
}
</style>
</head>
<body>

<table>
  <tr><th>Firstname</th><th>Lastname</th></tr>
  <tr><td>Peter</td><td>Griffin</td></tr>
  <tr><td>Lois</td><td>Griffin</td></tr>
</table>

</body>
</html>
```

Add a border to a table:

Firstname	Lastname
Peter	Griffin
Lois	Griffin

TABEL WIDTH

```
<!DOCTYPE html>
<html>
<head>
<style>
table, th, td {
  border: 1px solid;
}

table {
  width: 100%;
}
</style>
</head>
<body>
<h2>Full-width Table</h2>
<table>
  <tr><th>Firstname</th><th>Lastname</th></tr>
  <tr><td>Peter</td><td>Griffin</td></tr>
  <tr><td>Lois</td><td>Griffin</td></tr>
</table>

</body>
</html>
```

Full-width Table

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Collapse Table Borders

The `border-collapse` property sets whether the table borders should be collapsed into a single border:

```
<style>
table, td, th {
  border: 1px solid;
}

table {
  width: 100%;
  border-collapse: collapse;
}
</style>
</head>
<body>

<h2>Let the table borders collapse</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
</html>
```

Let the table borders collapse

Firstname	Lastname
Peter	Griffin
Lois	Griffin

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  width: 100%;
  border: 1px solid;
}
</style>
</head>
<body>

<h2>Single Border Around The Table</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
  </tr>
  <tr>
    <td>Lois</td>
    <td>Griffin</td>
  </tr>
</table>

</body>
</html>
```

Single Border Around The Table

Firstname	Lastname
Peter	Griffin
Lois	Griffin

Table

Table Width and Height

The width and height of a table are defined by the `width` and `height` properties.

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {border: 1px solid black;}
table {border-collapse: collapse;width: 100%;}
th {height: 50px;}
</style>
</head>
<body>

<h2>The width and height Properties</h2>
<p>Set the width of the table, and the height of the table header row:</p>

<table>
<tr><th>Firstname</th><th>Lastname</th></tr>
<tr><td>Peter</td><td>Griffin</td></tr>
<tr><td>Lois</td><td>Griffin</td></tr>
<tr><td>Joe</td><td>Swanson</td></tr>
<tr><td>Cleveland</td><td>Brown</td></tr>
</table>

</body>
</html>
```

The width and height Properties

Set the width of the table, and the height of the table header row:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

Table Alignment

Horizontal Alignment

The `text-align` property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>.

By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned.

To center-align the content of <td> elements as well, use `text-align: center`:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

```
    td {
  text-align: center;
}
```

To left-align the content, force the alignment of <th> elements to be left-aligned, with the `text-align: left` property:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
th {
  text-align: left;
}
```

Vertical Alignment

The `vertical-align` property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>.

By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).

The following example sets the vertical text alignment to bottom for <td> elements:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300

Example

```
td {
  height: 50px;
  vertical-align: bottom;
}
```

Table Style

Table Padding

To control the space between the border and the content in a table, use the `padding` property on <td> and <th> elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
table, td, th {
  border: 1px solid #ddd;
  text-align: left;
}

table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 10px;
}
</style>
</head>
<body>

<h2>The padding Property</h2>

<p>This property adds space between the border and the content in a table.</p>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
```

The padding Property

This property adds space between the border and the content in a table.

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

Horizontal Dividers

Add the `border-bottom` property to <th> and <td> for horizontal dividers:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}
</style>
</head>
<body>

<h2>Bordered Table Dividers</h2>
<p>Add the border-bottom property to th and td for horizontal dividers:</p>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
  <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
```

Bordered Table Dividers

Add the border-bottom property to th and td for horizontal dividers:

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

Hoverable Table

Use the `:hover` selector on <tr> to highlight table rows on mouse over:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  padding: 8px;
  text-align: left;
  border-bottom: 1px solid #ddd;
}

tr:hover {background-color: coral;}
</style>
</head>
<body>

<h2>Hoverable Table</h2>

<p>Move the mouse over the table rows to see the effect.</p>

<table>
  <tr>
    <th>First Name</th>
    <th>Last Name</th>
    <th>Points</th>
```

Hoverable Table

Move the mouse over the table rows to see the effect.

First Name	Last Name	Points
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

Striped Tables

For zebra-striped tables, use the `nth-child()` selector and add a `background-color` to all even (or odd) table rows:

```
<!DOCTYPE html>
<html>
<head>
<style>
table {
  border-collapse: collapse;
  width: 100%;
}

th, td {
  text-align: left;
  padding: 8px;
}

tr:nth-child(even) {background-color: #f2f2f2;}
</style>
</head>
<body>

<h2>Striped Table</h2>

<p>For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows:</p>

<table>
  <tr>
  <th>First Name</th>
  <th>Last Name</th>
  <th>Points</th>
```

Striped Table

For zebra-striped tables, use the nth-child() selector and add a background-color to all even (or odd) table rows:

First Name	Last Name	Points
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

```html
<!DOCTYPE html>
<html>
<head>
<style>
table {border-collapse: collapse;width: 100%;}
th, td {text-align: left;padding: 8px;}
tr:nth-child(even){background-color: #f2f2f2}
th {background-color: #04AA6D;color: white;}
</style>
</head>
<body>

<h2>Colored Table Header</h2>

<table>
  <tr>
    <th>Firstname</th>
    <th>Lastname</th>
    <th>Savings</th>
  </tr>
  <tr>
    <td>Peter</td>
    <td>Griffin</td>
    <td>$100</td>
  </tr>
  <tr>
```

Colored Table Header

Firstname	Lastname	Savings
Peter	Griffin	$100
Lois	Griffin	$150
Joe	Swanson	$300
Cleveland	Brown	$250

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content:

Add a container element (like) with `overflow-x:auto` around the <table> element to make it responsive:

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

<style>
table {border-collapse: collapse;width: 100%;}
th, td {text-align: left;padding: 8px;}
tr:nth-child(even) {background-color: #f2f2f2;}
</style>

</head>
<body>
<h2>Responsive Table</h2>
<p>A responsive table will display a horizontal scroll bar if the screen is too
small to display the full content. Resize the browser window to see the effect:</p>
<p>To create a responsive table, add a container element (like div) with
<strong>overflow-x:auto</strong> around the table element:</p>

<div style="overflow-x: auto;">
  <table>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
      <th>Points</th>
```

Responsive Table

A responsive table will display a horizontal scroll bar if the screen is too small to display the full content. Resize the browser window to see the effect:

To create a responsive table, add a container element (like div) with **overflow-x:auto** around the table element:

First Name	Last Name	Points	Points	Points	Points	Points	Points	Points	Points	Points	Points
Jill	Smith	50	50	50	50	50	50	50	50	50	50
Eve	Jackson	94	94	94	94	94	94	94	94	94	94
Adam	Johnson	67	67	67	67	67	67	67	67	67	67

DISPLAY

The display Property

The `display` property specifies if/how an element is displayed.

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

Block-level Elements

A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can).

The element is a block-level element.

Examples of block-level elements:

-
- <h1> - <h6>
- <p>
- <form>
- <header>
- <footer>
- <section>

Inline Elements

An inline element does not start on a new line and only takes up as much width as necessary.

This is an inline element inside a paragraph.

Examples of inline elements:

-
- <a>
- <img>

Display: none;

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

Take a look at our last example on this page if you want to know how this can be achieved.

The `<script>` element uses `display: none;` as default.

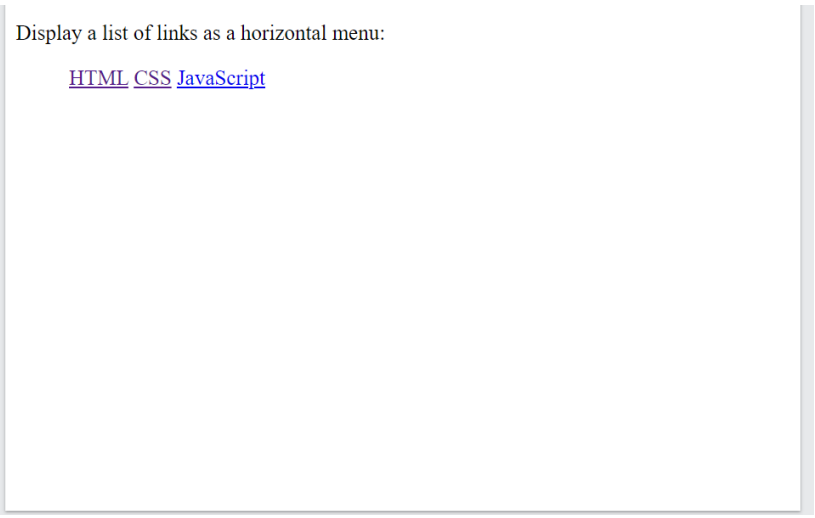
Override The Default Display Value

As mentioned, every element has a default display value. However, you can override this.

Changing an inline element to a block element, or vice versa, can be useful for making the page look a specific way, and still follow the web standards.

A common example is making inline `<li>` elements for horizontal menus:

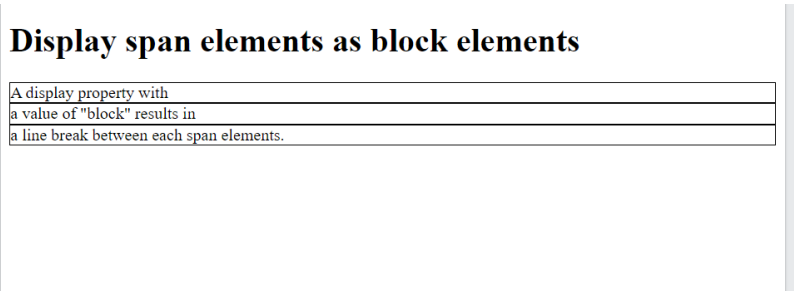
```
<!DOCTYPE html>
<html>
<head>
<style>
li {
  display: inline;
}
</style>
</head>
<body>

<p>Display a list of links as a horizontal menu:</p>

<ul>
  <li><a href="/html/default.asp" target="_blank">HTML</a></li>
  <li><a href="/css/default.asp" target="_blank">CSS</a></li>
  <li><a href="/js/default.asp" target="_blank">JavaScript</a></li>
</ul>

</body>
</html>
```



Note: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with `display: block;` is not allowed to have other block elements inside it.

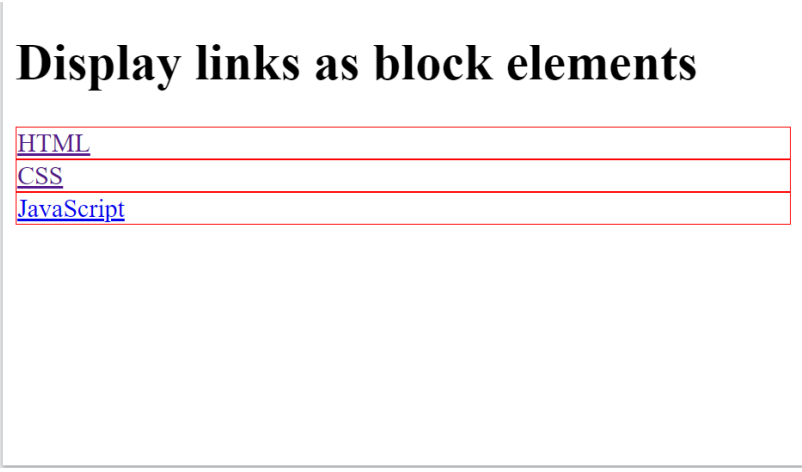
The following example displays elements as block elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
span {
  display: block; BORDER:1PX SOLID
}
</style>
</head>
<body>
<h1>Display span elements as block elements</h1>
<span>A display property with</span> <span>a value of "block" results in</span> <span>a
line break between each span elements.</span>
</body>
</html>
```



The following example displays <a> elements as block elements:

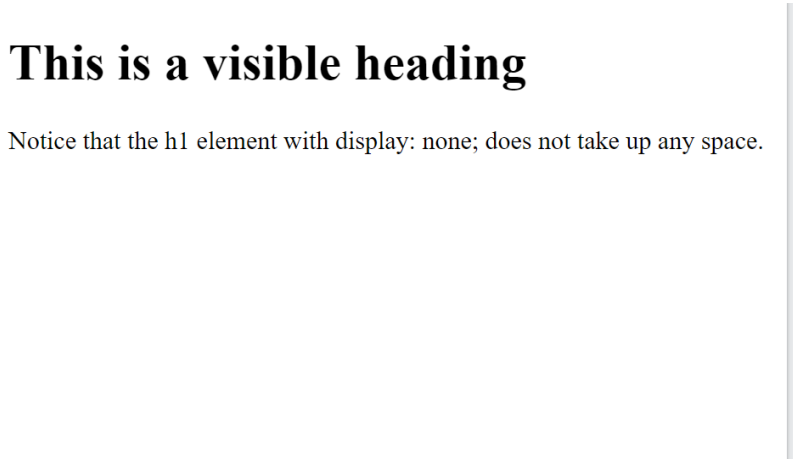
```
<!DOCTYPE html>
<html>
<head>
<style>
a {
  display: block;BORDER:1PX SOLID RED
}
</style>
</head>
<body>
<h1>Display links as block elements</h1>
<a href="/html/default.asp" target="_blank">HTML</a>
<a href="/css/default.asp" target="_blank">CSS</a>
<a href="/js/default.asp" target="_blank">JavaScript</a>
</body>
</html>
```



Hide an Element - display:none or visibility:hidden?

Hiding an element can be done by setting the `display` property to `none`. The element will be hidden, and the page will be displayed as if the element is not there:

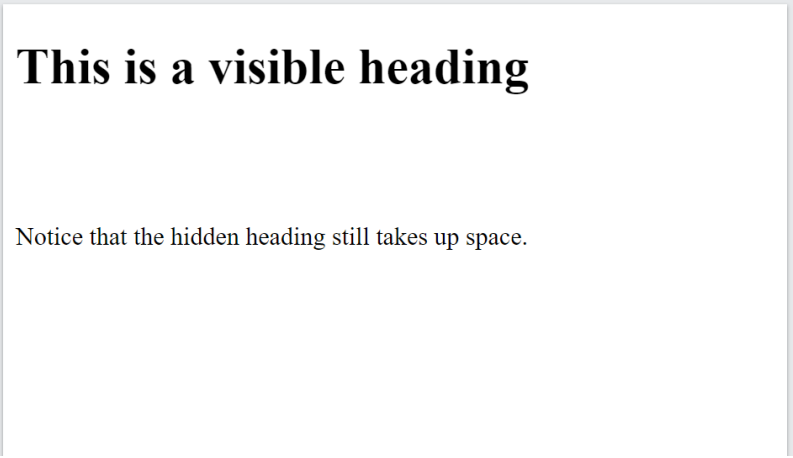
```
<!DOCTYPE html>
<html>
<head>
<style>
h1.hidden {
  display: none;
}
</style>
</head>
<body>
<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the h1 element with display: none; does
not take up any space.</p>
</body>
</html>
```



`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:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1.hidden {
  visibility: hidden;
}
</style>
</head>
<body>
<h1>This is a visible heading</h1>
<h1 class="hidden">This is a hidden heading</h1>
<p>Notice that the hidden heading still takes up space.
</p>
</body>
</html>
```



Layout - width and max-width

Using width, max-width and margin: auto;

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the `width` of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins:

This element has a width of 500px, and margin set to auto.

Note: The problem with the `<div>` above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

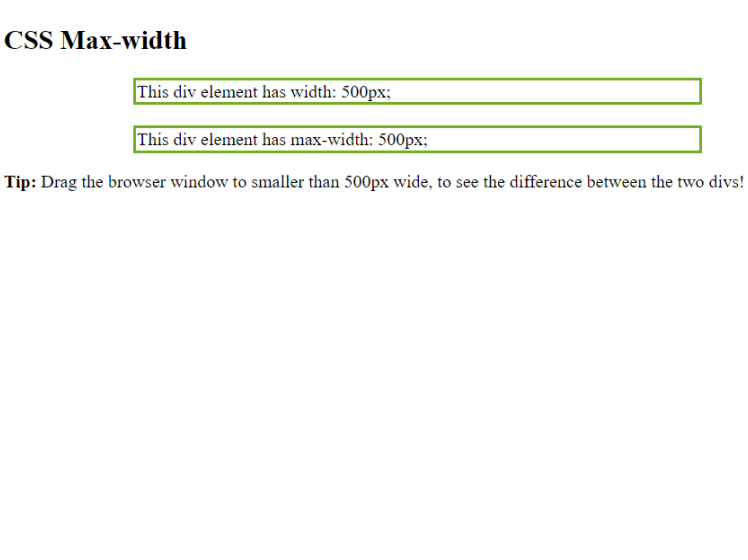
Using `max-width` instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices:

This element has a max-width of 500px, and margin set to auto.

Tip: Resize the browser window to less than 500px wide, to see the difference between the two divs!

Here is an example of the two divs above:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
  width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}

div.ex2 {
  max-width: 500px;
  margin: auto;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>CSS Max-width</h2>
<div class="ex1">This div element has width: 500px;</div>
<br>
<div class="ex2">This div element has max-width: 500px;</div>
<p><strong>Tip:</strong> Drag the browser window to smaller than 500px wide, to see the
difference between
the two divs!</p>
</body>
</html>
```



POSITION

The position Property

The `position` property specifies the type of positioning method used for an element.

There are five different position values:

- `static`
- `relative`
- `fixed`
- `absolute`
- `sticky`

Elements are then positioned using the top, bottom, left, and right properties.

However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

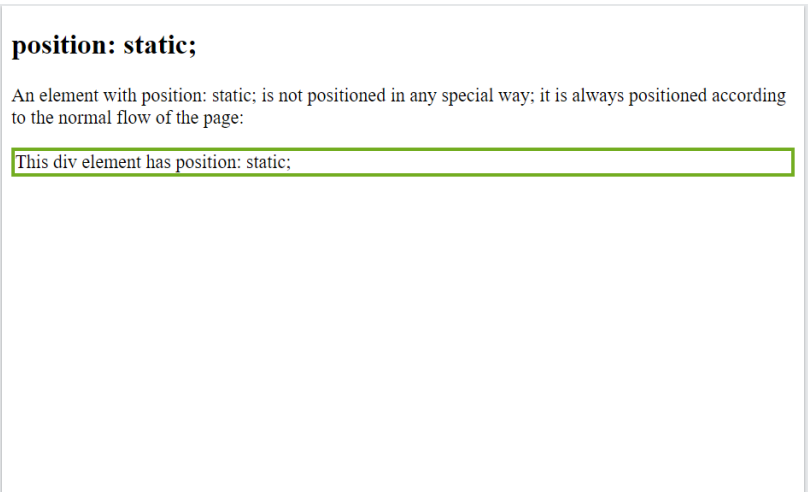
position: static;

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:

```
<!DOCTYPE html>
<html>
<head>
<style>
div.static {
  position: static;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>

<h2>position: static;</h2>

<p>An element with position: static; is not positioned in any special way; it
is always positioned according to the normal flow of the page:</p>

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

</body>
</html>
```



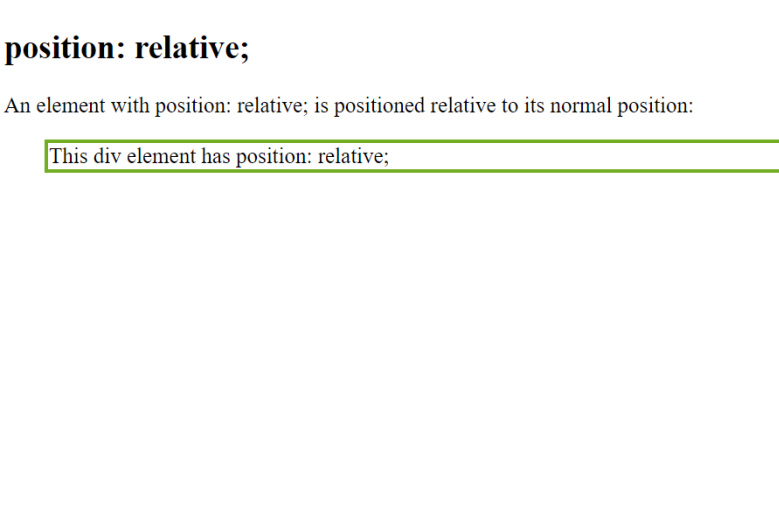
position: relative;

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.

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  left: 30px;
  border: 3px solid #73AD21;=}
</style>
</head>
<body>
<h2>position: relative;</h2>
<p>An element with position: relative; is positioned relative to its
normal position:</p>
<div class="relative">
This div element has position: relative;
</div>

</body>
</html>
```



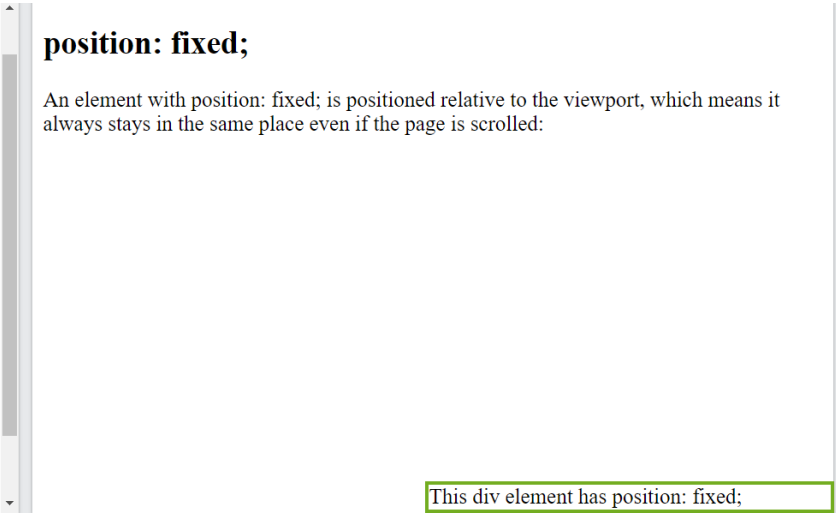
position: fixed;

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.

Notice the fixed element in the lower-right corner of the page. Here is the CSS that is used:

```
<head>
<style>
div.fixed {
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
</style>
</head>
<body>
<h2>position: fixed;</h2>
<p>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:</p>
<div class="fixed">
This div element has position: fixed;
</div>
</body>
</html>
```



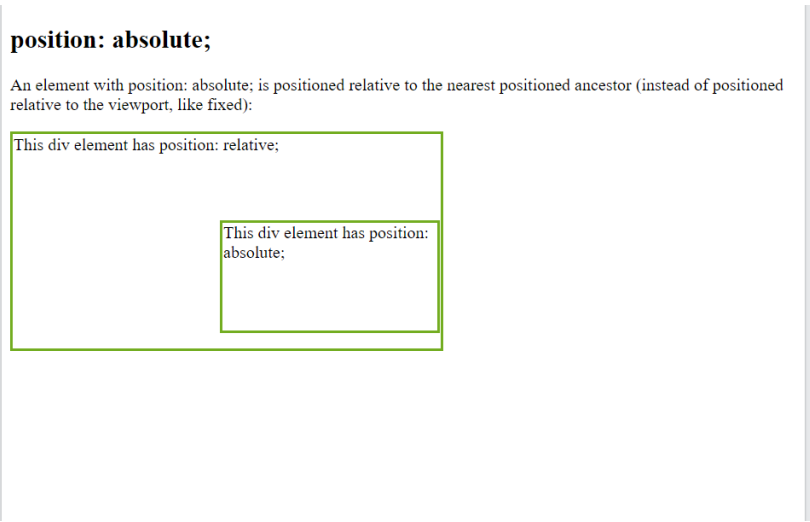
position: absolute;

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.

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

```
<!DOCTYPE html>
<html>
<head>
<style>
div.relative {
  position: relative;
  width: 400px;
  height: 200px;
  border: 3px solid #73AD21;}
div.absolute {
  position: absolute;
  top: 80px;
  right: 0;
  width: 200px;
  height: 100px;
  border: 3px solid #73AD21;}
</style>
</head>
<body>
<h2>position: absolute;</h2>
<p>An element with position: absolute; is positioned relative to the nearest positioned
ancestor (instead of positioned relative to the viewport, like fixed):</p>
<div class="relative">This div element has position: relative;
  <div class="absolute">This div element has position: absolute;</div>
</div>
</body>
</html>
```

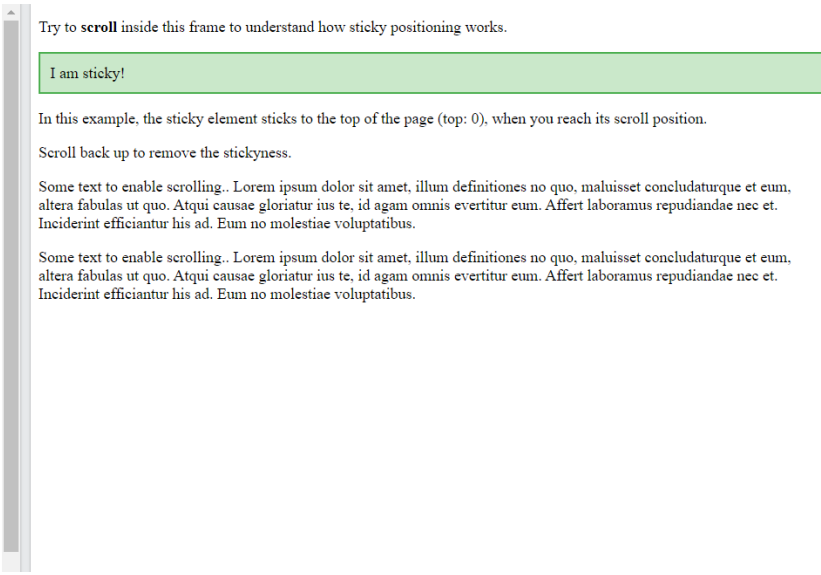


position: sticky;

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).

```
<!DOCTYPE html>
<html>
<head>
<style>
div.sticky {
  position: -webkit-sticky;
  position: sticky;
  top: 0;
  padding: 10px;
  background-color: #cae8ca;
  border: 2px solid #4CAF50;}
</style>
</head>
<body>
<p>Try to <b>scroll</b> inside this frame to understand how sticky positioning works.</p>
<div class="sticky">I am sticky!</div>
<div style="padding-bottom:2000px">
  <p>In this example, the sticky element sticks to the top of the page (top: 0), when you reach its scroll position.</p>
  <p>Scroll back up to remove the stickyness.</p>
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
  <p>Some text to enable scrolling.. Lorem ipsum dolor sit amet, illum definitiones no quo, maluisset concludaturque et eum, altera fabulas ut quo. Atqui causae gloriatur ius te, id agam omnis evertitur eum. Affert laboramus repudiandae nec et. Inciderint efficiantur his ad. Eum no molestiae voluptatibus.</p>
</div>

</body>
</html>
```



Z-INDEX

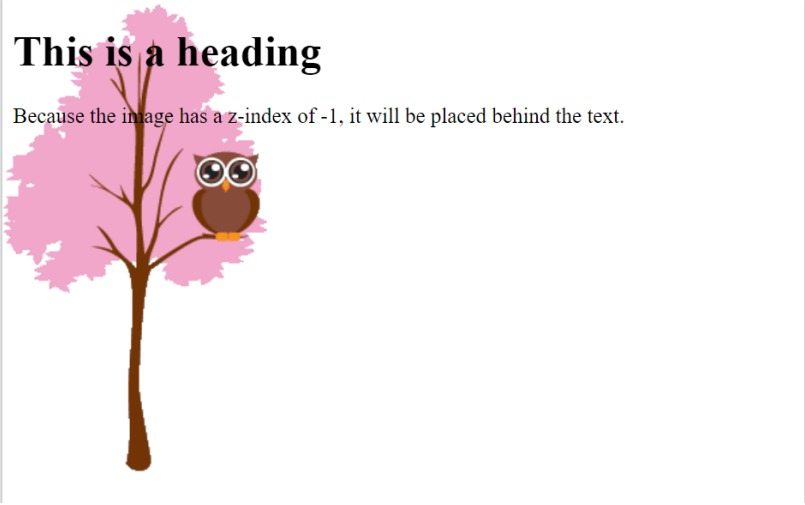
The z-index Property

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:

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<img src="img_tree.png">
<p>Because the image has a z-index of -1, it will be placed behind
the text.</p>

</body>
</html>
```



Note: `z-index` only works on positioned elements (position: absolute, position: relative, position: fixed, or position: sticky) and flex items (elements that are direct children of display: flex elements).

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:

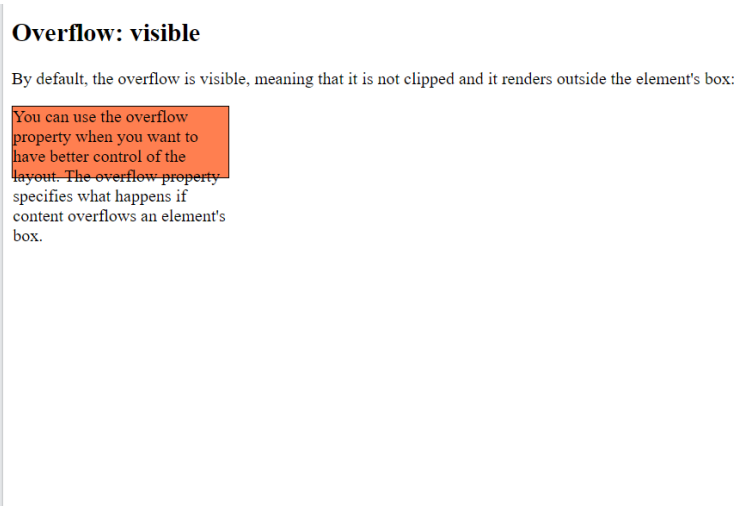
- `visible` - Default. The overflow is not clipped. The content renders outside the element's box
- `hidden` - The overflow is clipped, and the rest of the content will be invisible
- `scroll` - The overflow is clipped, and a scrollbar is added to see the rest of the content
- `auto` - Similar to `scroll`,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

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

overflow: visible

By default, the overflow is `visible`, meaning that it is not clipped and it renders outside the element's box:

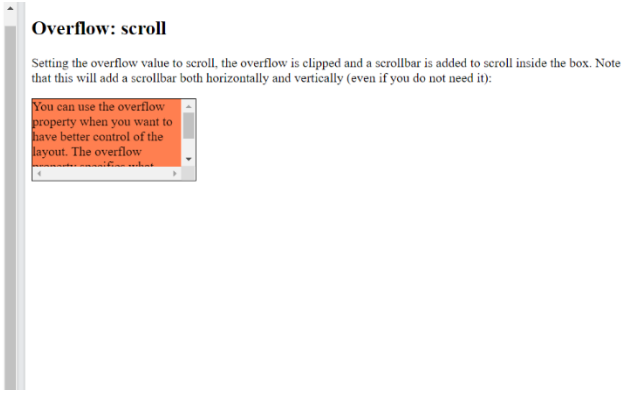
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: coral;
  width: 200px;
  height: 65px;
  border: 1px solid;
  overflow: visible;
}
</style>
</head>
<body>

<h2>Overflow: visible</h2>

<p>By default, the overflow is visible, meaning that it is not clipped and it renders
outside the element's box:</p>

<div>You can use the overflow property when you want to have better control of the
layout. The overflow property specifies what happens if content overflows an element's
box.</div>

</body>
</html>
```



overflow: scroll

Setting the value to `scroll`, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):

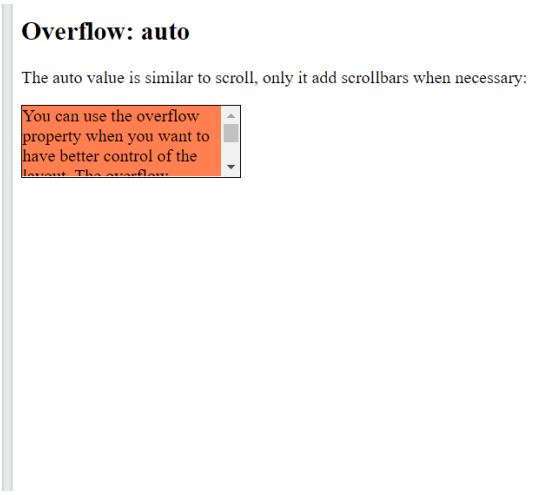
```
<html>
<head>
<style>
div {
  background-color: coral;
  width: 200px;
  height: 100px;
  border: 1px solid black;
  overflow: scroll;
}
</style>
</head>
<body>

<h2>Overflow: scroll</h2>

<p>Setting the overflow value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```



overflow: auto

The `auto` value is similar to `scroll`, but it adds scrollbars only when necessary:

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: coral;
  width: 200px;
  height: 65px;
  border: 1px solid black;
  overflow: auto;
}
</style>
</head>
<body>

<h2>Overflow: auto</h2>

<p>The auto value is similar to scroll, only it add scrollbars when necessary:</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```



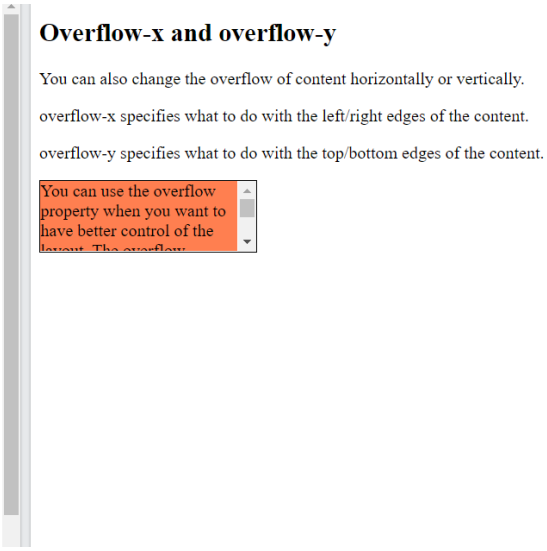
overflow-x and overflow-y

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

`overflow-x` specifies what to do with the left/right edges of the content.
`overflow-y` specifies what to do with the top/bottom edges of the content.

```
div {
  overflow-x: hidden; /* Hide horizontal scrollbar */
  overflow-y: scroll; /* Add vertical scrollbar */
}
```

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: coral;
  width: 200px;
  height: 65px;
  border: 1px solid black;
  overflow-x: hidden;
  overflow-y: scroll;
}
</style>
</head>
<body>

<h2>Overflow-x and overflow-y</h2>

<p>You can also change the overflow of content horizontally or vertically.</p>
<p>overflow-x specifies what to do with the left/right edges of the content.</p>
<p>overflow-y specifies what to do with the top/bottom edges of the content.</p>

<div>You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.</div>

</body>
</html>
```



FLOAT AND CLEAR

The CSS `float` property specifies how an element should float.

The CSS `clear` property specifies what elements can float beside the cleared element and on which side.

The float Property

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

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: right;
}
</style>
</head>
<body>

<h2>Float Right</h2>

<p>In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-left:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>
```

Float Right

In this example, the image will float to the right in the paragraph, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: left;
}
</style>
</head>
<body>

<h2>Float Left</h2>

<p>In this example, the image will float to the left in the paragraph, and the text in the paragraph will wrap around the image.</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;margin-right:15px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>
```

Float Left

In this example, the image will float to the left in the paragraph, and the text in the paragraph will wrap around the image.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

```
<!DOCTYPE html>
<html>
<head>
<style>
img {
  float: none;
}
</style>
</head>
<body>

<h2>Float None</h2>

<p>In this example, the image will be displayed just where it occurs in the text (float: none;).</p>

<p><img src="pineapple.jpg" alt="Pineapple" style="width:170px;height:170px;">
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.</p>

</body>
</html>
```

Float None

In this example, the image will be displayed just where it occurs in the text (float: none;).

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictun nisi, sed ullamcorper ipsum dignissim ac. In at libero sed nunc venenatis imperdiet sed ornare turpis. Donec vitae dui eget tellus gravida venenatis. Integer fringilla congue eros non fermentum. Sed dapibus pulvinar nibh tempor porta. Cras ac leo purus. Mauris quis diam velit.

The clear Property

When we use the `float` property, and we want the next element below (not on right or left), we will have to use the `clear` property.

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:

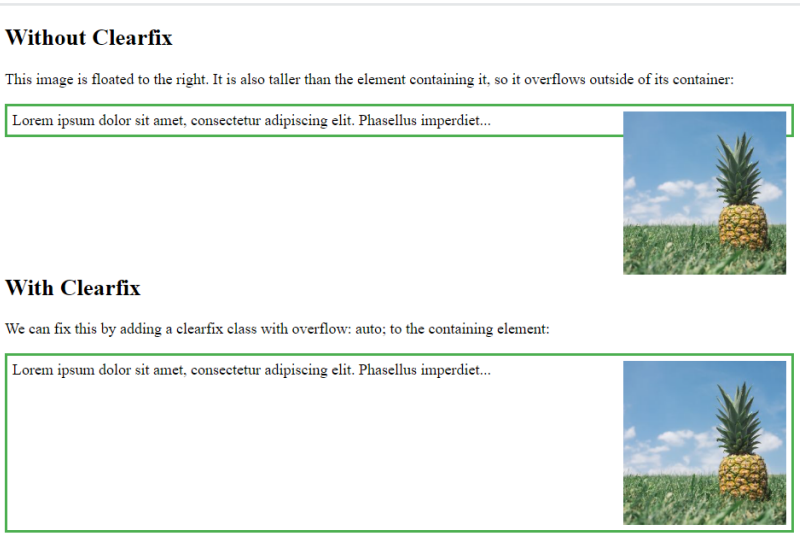
- `none` - The element is not pushed below left or right floated elements. This is default
- `left` - The element is pushed below left floated elements
- `right` - The element is pushed below right floated elements
- `both` - The element is pushed below both left and right floated elements
- `inherit`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

```
<!DOCTYPE html><html><head><style>
.div1 {
  float: left;
  padding: 10px;
  border: 3px solid #73AD21;}
.div2 {
  padding: 10px;
  border: 3px solid red;}
.div3 {
  float: left;
  padding: 10px;
  border: 3px solid #73AD21;}
.div4 {
  padding: 10px;
  border: 3px solid red;
  clear: left;}
</style>
</head>
<body>
<h2>Without clear</h2>
<div class="div1">div1</div>
<div class="div2">div2 - Notice that div2 is after div1 in the HTML code. However,
since div1 floats to the left, the text in div2 flows around div1.</div>
<br><br>
<h2>With clear</h2>
<div class="div3">div3</div>
<div class="div4">div4 - Here, clear: left; moves div4 down below the floating div3.
The value "left" clears elements floated to the left. You can also clear "right" and
"both".</div></body></html>
```

The clearfix Hack

If a floated element is taller than the containing element, it will "overflow" outside of its container. We can then add a clearfix hack to solve this problem:

```
<!DOCTYPE html><html><head><style>
div {
  border: 3px solid #4CAF50;
  padding: 5px;}
.img1 {
  float: right;}
.img2 {
  float: right;}
.clearfix {
  overflow: auto;}

</style></head><body>
<h2>Without Clearfix</h2>

<p>This image is floated to the right. It is also taller than the element containing it, so it
overflows outside of its container:</p>

<div>
  <img class="img1" src="pineapple.jpg" alt="Pineapple" width="170" height="170">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...
</div>

<h2 style="clear:right">With Clearfix</h2>
<p>We can fix this by adding a clearfix class with overflow: auto; to the containing element:</p>

<div class="clearfix">
  <img class="img2" src="pineapple.jpg" alt="Pineapple" width="170" height="170">
  Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet...
</div></body></html>
```



The `overflow: auto` clearfix works well as long as you are able to keep control of your margins and padding (else you might see scrollbars). The **new, modern clearfix hack** however, is safer to use, and the following code is used for most webpages:

INLINE-BLOCK

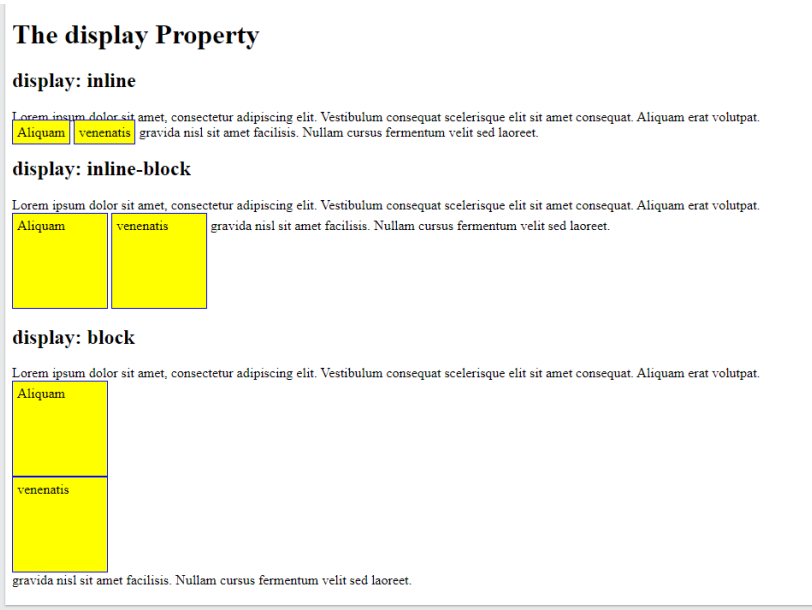
Compared to `display: inline`, the major difference is that `display: inline-block` allows to set a width and height on the element.

Also, with `display: inline-block`, the top and bottom margins/paddings are respected, but with `display: inline` they are not.

Compared to `display: block`, the major difference is that `display: inline-block` does not add a line-break after the element, so the element can sit next to other elements.

The following example shows the different behavior of `display: inline`, `display: inline-block` and `display: block`:

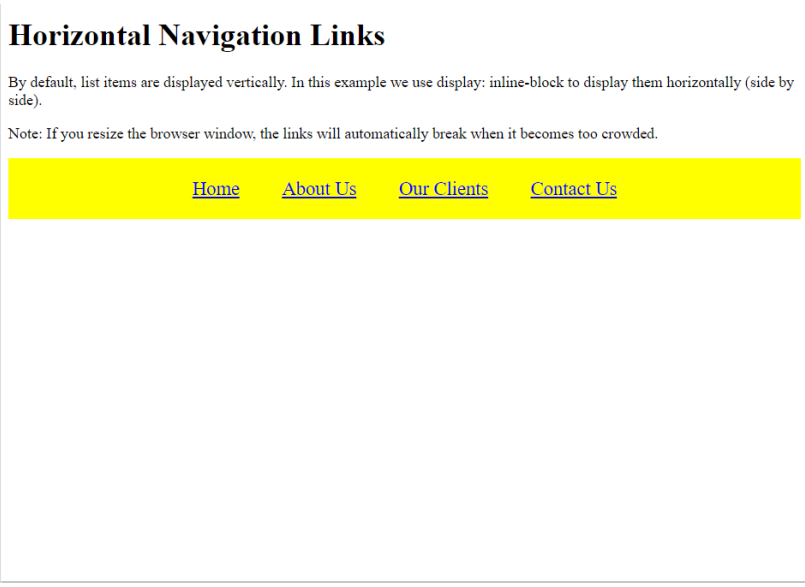
```
<!DOCTYPE html><html><head><style>
span.a {
  display: inline; /* the default for span */
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow; }
span.b {
  display: inline-block;
  width: 100px;
  height: 100px;
  padding: 5px;
  border: 1px solid blue;
  background-color: yellow; }
span.c {display: block;width: 100px;height: 100px;padding: 5px;border: 1px solid blue;    background-color:
yellow; }
</style></head>body>

<h1>The display Property</h1>

<h2>display: inline</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet
consequat. Aliquam erat volutpat. <span class="a">Aliquam</span> <span class="a">venenatis</span> gravida nisl
sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

<h2>display: inline-block</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet
consequat. Aliquam erat volutpat. <span class="b">Aliquam</span> <span class="b">venenatis</span> gravida nisl
sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

<h2>display: block</h2>
<div>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet
consequat. Aliquam erat volutpat. <span class="c">Aliquam</span> <span class="c">venenatis</span> gravida nisl
sit amet facilisis. Nullam cursus fermentum velit sed laoreet. </div>

</body>
</html>
```



Using inline-block to Create Navigation Links

One common use for `display: inline-block` is to display list items horizontally instead of vertically. The following example creates horizontal navigation links:

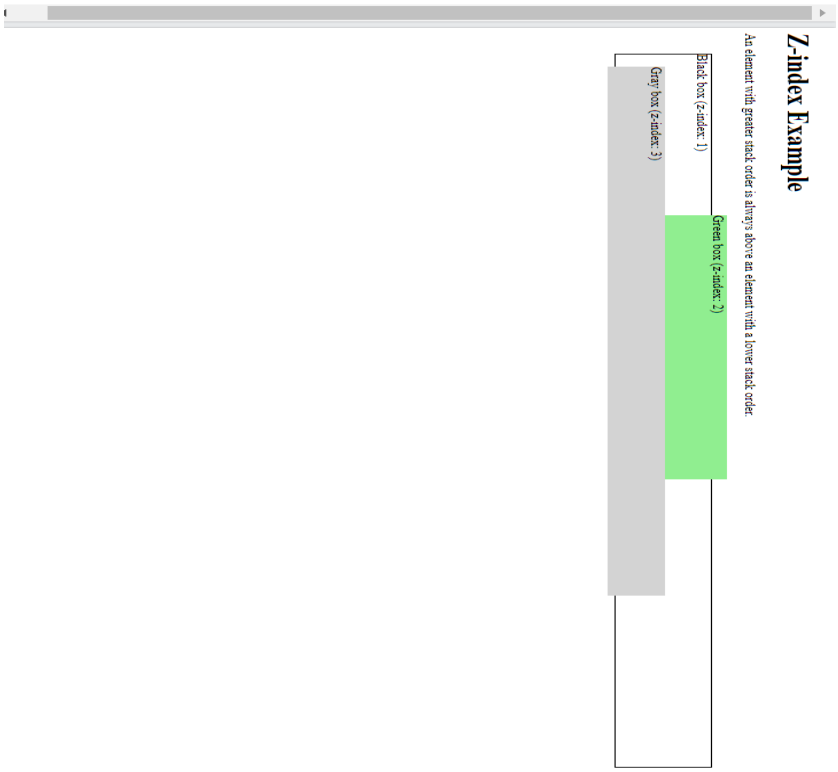
```
<head>
<style>
.nav {
  background-color: yellow;
  list-style-type: none;
  text-align: center;
  margin: 0;
  padding: 0;
}

.nav li {
  display: inline-block;
  font-size: 20px;
  padding: 20px;
}
</style>
</head>
<body>

<h1>Horizontal Navigation Links</h1>
<p>By default, list items are displayed vertically. In this example we use display: inline-block
to display them horizontally (side by side).</p>
<p>Note: If you resize the browser window, the links will automatically break when it becomes too
crowded.</p>

<ul class="nav">
  <li><a href="#home">Home</a></li>
  <li><a href="#about">About Us</a></li>
  <li><a href="#clients">Our Clients</a></li>
  <li><a href="#contact">Contact Us</a></li>
</ul>

</body>
</html>
```



Z-INDEX

Here we see that an element with greater stack order is always above an element with a lower stack order:

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

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

.gray-box {
  position: absolute;
  z-index: 3; /* gray box will be above both green and black box */
  background: lightgray;
  height: 60px;
  width: 70%;
  left: 50px;
  top: 50px;
}

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

<h1>Z-index Example</h1>

<p>An element with greater stack order is always above an element with a lower stack order.</p>

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

</body>
</html>
```



Combinators Selector

A combinator is something that explains the relationship between the selectors.

A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator.

There are four different combinators in CSS:

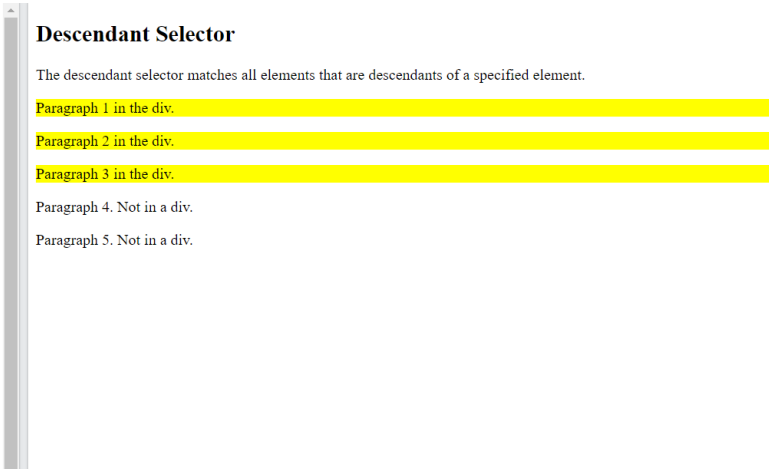
- descendant selector (space)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

Descendant Selector

The descendant selector matches all elements that are descendants of a specified element.

The following example selects all <p> elements inside elements:

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

<h2>Descendant Selector</h2>

<p>The descendant selector matches all elements that are descendants of a specified element.</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section><p>Paragraph 3 in the div.</p></section>
</div>

<p>Paragraph 4. Not in a div.</p>
<p>Paragraph 5. Not in a div.</p>

</body>
</html>
```

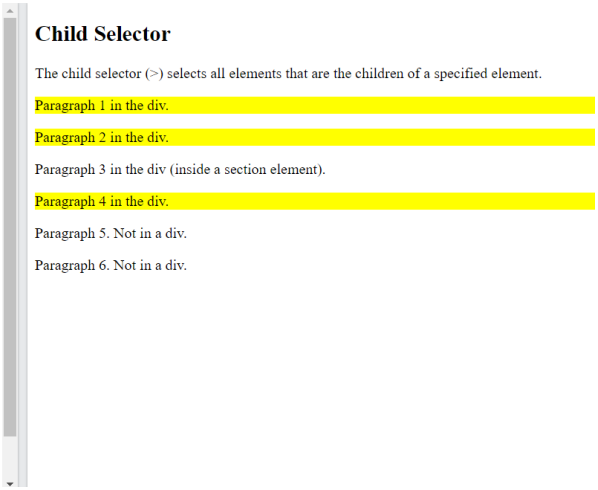


Child Selector (>)

The child selector selects all elements that are the children of a specified element.

The following example selects all <p> elements that are children of a element:

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

<h2>Child Selector</h2>

<p>The child selector (>) selects all elements that are the children of a specified element.</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
  <section>
    <!-- not Child but Descendant -->
    <p>Paragraph 3 in the div (inside a section element).</p>
  </section>
  <p>Paragraph 4 in the div.</p>
</div>
<p>Paragraph 5. Not in a div.</p>
<p>Paragraph 6. Not in a div.</p>
</body>
</html>
```



Adjacent Sibling Selector (+)

The adjacent sibling selector is used to select an element that is directly after another specific element.

Sibling elements must have the same parent element, and "adjacent" means "immediately following".

The following example selects the first <p> element that are placed immediately after elements:

```html
<!DOCTYPE html>
<html>
<head> <style>
div + p {
  background-color: yellow;
}
</style>
</head>
<body>
<h2>Adjacent Sibling Selector</h2>
<p>The + selector is used to select an element that is directly after another specific element.</p>
<p>The following example selects the first p element that are placed immediately after div elements:</p>

<div>
  <p>Paragraph 1 in the div.</p>
  <p>Paragraph 2 in the div.</p>
</div>
<p>Paragraph 3. After a div.</p>
<p>Paragraph 4. After a div.</p>
<div>
  <p>Paragraph 5 in the div.</p>
  <p>Paragraph 6 in the div.</p>
</div>
<p>Paragraph 7. After a div.</p>
<p>Paragraph 8. After a div.</p>

</body></html>
```

Adjacent Sibling Selector

The + selector is used to select an element that is directly after another specific element.

The following example selects the first p element that are placed immediately after div elements:

Paragraph 1 in the div.

Paragraph 2 in the div.

Paragraph 3. After a div.

Paragraph 4. After a div.

Paragraph 5 in the div.

Paragraph 6 in the div.

Paragraph 7. After a div.

Paragraph 8. After a div.

General Sibling Selector (~)

The general sibling selector selects all elements that are next siblings of a specified element.

The following example selects all <p> elements that are next siblings of elements:

```html
<!DOCTYPE html>
<html>
<head>
<style>
div ~ p {
  background-color: yellow;
}
</style>
</head>
<body>

<h2>General Sibling Selector</h2>

<p>The general sibling selector (~) selects all elements that are next siblings of a specified element.</p>

<p>Paragraph 1.</p>

<div>
  <p>Paragraph 2.</p>
</div>

<p>Paragraph 3.</p>
<code>Some code.</code>
<p>Paragraph 4.</p>

</body>
</html>
```

General Sibling Selector

The general sibling selector (~) selects all elements that are next siblings of a specified element.

Paragraph 1.

Paragraph 2.

Paragraph 3.

`Some code.`

Paragraph 4.

Pseudo-classes

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

For example, it can be used to:

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

Syntax

The syntax of pseudo-classes:

```
selector:pseudo-class {
  property: value;
}
```

Anchor Pseudo-classes

Links can be displayed in different ways:

```
/* unvisited link */
a:link {
  color: #FF0000; }


/* visited link */
a:visited {
  color: #00FF00;}


/* mouse over link */
a:hover {
  color: #FF00FF; }

/* selected link */
a:active {
  color: #0000FF; }
```

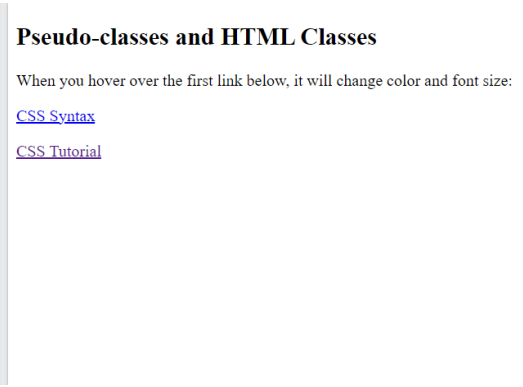
Note: `a:hover` MUST come after `a:link` and `a:visited` in the CSS definition in order to be effective! `a:active` MUST come after `a:hover` in the CSS definition in order to be effective! Pseudo-class names are not case-sensitive.

Pseudo-classes and HTML Classes

Pseudo-classes can be combined with HTML classes:

When you hover over the link in the example, it will change color:

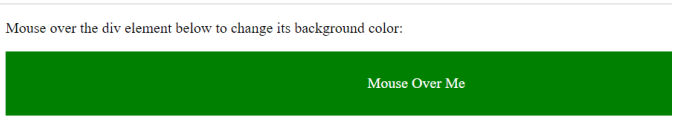
```
<!DOCTYPE html>
<html>
<head>
<style>
a.highlight:hover {
  color: red;
  font-size: 22px;
}
</style>
</head>
<body>
<h2>Pseudo-classes and HTML Classes</h2>

<p>When you hover over the first link below, it will change color and font size:</p>

<p><a class="highlight" href="css_syntax.asp">CSS Syntax</a></p>

<p><a href="default.asp">CSS Tutorial</a></p>
</body>
</html>
```



Hover on

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

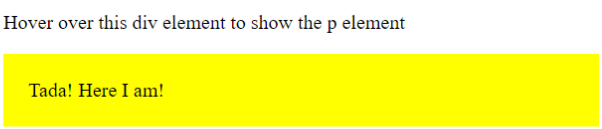
```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  background-color: green;
  color: white;
  padding: 25px;
  text-align: center;
}

div:hover {
  background-color: blue;
}
</style>
</head>
<body>

<p>Mouse over the div element below to change its background color:</p>

<div>Mouse Over Me</div>

</body>
</html>
```



Simple Tooltip Hover

Hover over a element to show a <p> element (like a tooltip):

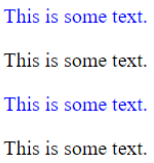
```
<!DOCTYPE html>
<html>
<head>
<style>
p {
  display: none;
  background-color: yellow;
  padding: 20px;
}

div:hover p {
  display: block;
}
</style>
</head>
<body>

<div>Hover over this div element to show the p element
  <p>Tada! Here I am!</p>
</div>

</body>
</html>
```



:first-child Pseudo-class

The `:first-child` pseudo-class matches a specified element that is the first child of another element.

```
<!DOCTYPE html>
<html>
<head>
<style>
p:first-child {
  color: blue;
}
</style>
</head>
<body>

<p>This is some text.</p>
<p>This is some text.</p>

<div>
  <p>This is some text.</p>
  <p>This is some text.</p>
</div>

</body>
</html>
```



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

In the following example, the selector matches the first <i> element in all <p> elements:

```
p i:first-child {
  color: blue;
}
```

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

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element:

```
p:first-child i {
  color: blue;
}
```

All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent
:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	p:nth-last-child(2)	Selects every <p> element that is the second child of its parent, counting from the last child
:nth-last-of-type(n)	p:nth-last-of-type(2)	Selects every <p> element that is the second <p> element of its parent, counting from the last child
:nth-of-type(n)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent

:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

Pseudo-Elements

A CSS pseudo-element is used to style specified parts of an element.

For example, it can be used to:

- Style the first letter, or line, of an element
- Insert content before, or after, the content of an element

Syntax

The syntax of pseudo-elements:

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

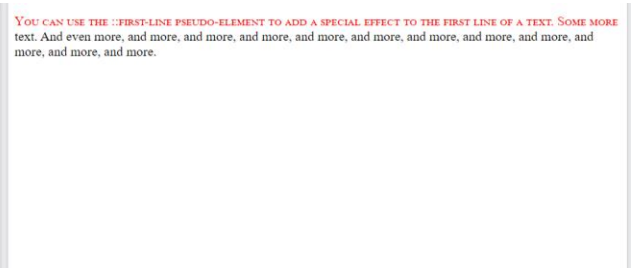
The ::first-line Pseudo-element

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

The following example formats the first line of the text in all <p> elements:

It change the color of first line .

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
</style>
</head>
<body>

<p>You can use the ::first-line pseudo-element to add a special effect to the first line of a text. Some more text. And even more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more, and more.</p>

</body>
</html>
```



Note: The `::first-line` pseudo-element can only be applied to block-level elements.

The following properties apply to the `::first-line` pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

Notice the double colon notation - `::first-line` versus `:first-line`

The double colon replaced the single-colon notation for pseudo-elements in CSS3. This was an attempt from W3C to distinguish between **pseudo-classes** and **pseudo-elements**.

The single-colon syntax was used for both pseudo-classes and pseudo-elements in CSS2 and CSS1.

For backward compatibility, the single-colon syntax is acceptable for CSS2 and CSS1 pseudo-elements.

::first-letter Pseudo-element

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

The following example formats the first letter of the text in all <p> elements:

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
</style>
</head>
<body>

<p>You can use the ::first-letter pseudo-element to add a special effect to the first character of a text!</p>

</body>
</html>
```

You can use the ::first-letter pseudo-element to add a special effect to the first character of a text!

Note: The `::first-letter` pseudo-element can only be applied to block-level elements.

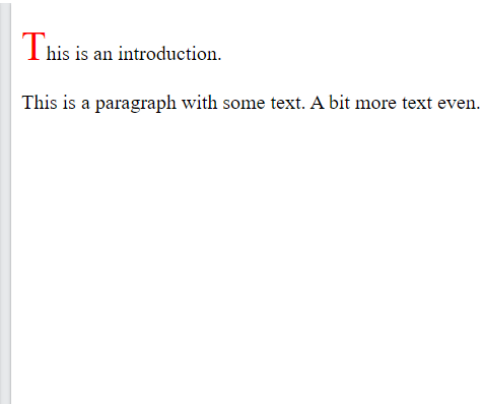
The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

Pseudo-elements and HTML Classes

Pseudo-elements can be combined with HTML classes:

```
<!DOCTYPE html>
<html>
<head>
<style>
p.intro::first-letter {
  color: #ff0000;
  font-size: 200%;
}
</style>
</head>
<body>

<p class="intro">This is an introduction.</p>
<p>This is a paragraph with some text. A bit more text even.</p>

</body>
</html>
```



The example above will display the first letter of paragraphs with class="intro", in red and in a larger size.

Multiple Pseudo-elements

Several pseudo-elements can also be combined.

In the following example, the first letter of a paragraph will be red, in an xx-large font size. The rest of the first line will be blue, and in small-caps. The rest of the paragraph will be the default font size and color:

```
<!DOCTYPE html>
<html>
<head>
<style>
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}

p::first-line {
  color: #0000ff;
  font-variant: small-caps;
}
</style>
</head>
<body>

<p>You can combine the ::first-letter and ::first-line pseudo-elements to add a special
effect to the first letter and the first line of a text!</p>

</body>
</html>
```

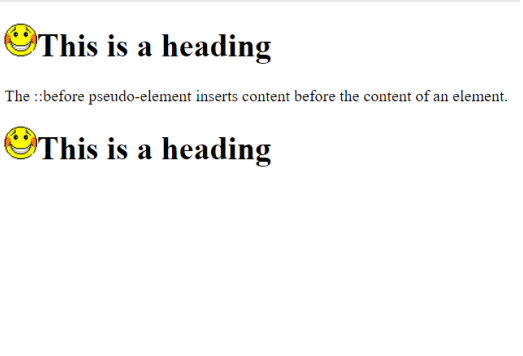
You can combine the ::first-letter and ::first-line pseudo-elements to add a special effect to the first letter and the first line of a text!

::before Pseudo-element

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

The following example inserts an image before the content of each <h1> element:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1::before {
  content: url(smiley.gif);
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>The ::before pseudo-element inserts content before the content of an element.
</p>

<h1>This is a heading</h1>

</body>
</html>
```

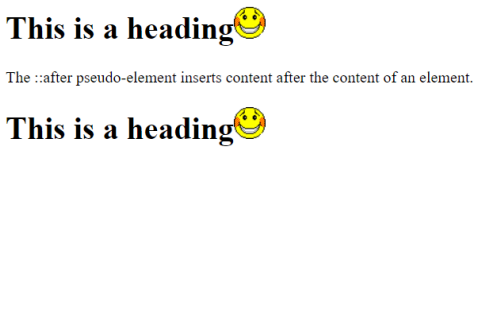


::after Pseudo-element

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

The following example inserts an image after the content of each <h1> element:

```
<!DOCTYPE html>
<html>
<head>
<style>
h1::after {
  content: url(smiley.gif);
}
</style>
</head>
<body>

<h1>This is a heading</h1>
<p>The ::after pseudo-element inserts content after the content of an element.</p>

<h1>This is a heading</h1>

</body>
</html>
```

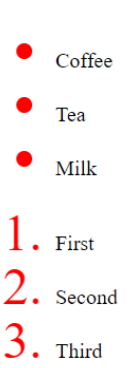


::marker Pseudo-element

The `::marker` pseudo-element selects the markers of list items.

The following example styles the markers of list items:

```
<!DOCTYPE html>
<html>
<head>
<style>
::marker {
  color: red;
  font-size:40px;
}
</style>
</head>
<body>

<ul>
  <li>Coffee</li>
  <li>Tea</li>
  <li>Milk</li>
</ul>

<ol>
  <li>First</li>
  <li>Second</li>
  <li>Third</li>
</ol>

</body>
</html>
```



::selection Pseudo-element

The `::selection` pseudo-element matches the portion of an element that is selected by a user.

The following CSS properties can be applied to `::selection`: `color`, `background`, `cursor`, and `outline`.

The following example makes the selected text red on a yellow background:

```
::selection {
  color: red;
  background: yellow;
}
```

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::marker	::marker	Selects the markers of list items
::selection	p::selection	Selects the portion of an element that is selected by a user

Attribute Selector

Selector	Example	Example description
[*attribute*]	[target]	Selects all elements with a target attribute
[*attribute=value*]	[target="_blank"]	Selects all elements with target="_blank"
[*attribute~=value*]	[title~="flower"]	Selects all elements with a title attribute containing the word "flower"
[*attribute|=value*]	[lang|="en"]	Selects all elements with a lang attribute value starting with "en"
[*attribute^=value*]	a[href^="https"]	Selects every <a> element whose href attribute value begins with "https"
[*attribute$=value*]	a[href$=".pdf"]	Selects every <a> element whose href attribute value ends with ".pdf"
[*attribute*=value*]	a[href*="w3schools"]	Selects every <a> element whose href attribute value contains the substring "w3schools"

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

[attribute] Selector

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

The following example selects all <a> elements with a target attribute:

```
a[target] {
  background-color: yellow;
}
```

[attribute="value"] Selector

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

The following example selects all <a> elements with a target="_blank" attribute:

```
a[target="_blank"] {
  background-color: yellow;
}
```

[attribute~="value"] Selector

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

The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

```
[title~="flower"] {
  border: 5px solid yellow;
}
```

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

[attribute|="value"] Selector

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

Note: The value has to be a whole word, either alone, like class="top", or followed by a hyphen(-), like class="top-text".

```
[class|="top"] {
  background: yellow;
}
```

[attribute^="value"] Selector

The `[attribute^="value"]` selector is used to select elements with the specified attribute, whose value starts with the specified value.

The following example selects all elements with a class attribute value that starts with "top":

Note: The value does not have to be a whole word!

```
[class^="top"] {
  background: yellow;
}
```

[attribute$="value"] Selector

The `[attribute$="value"]` selector is used to select elements whose attribute value ends with a specified value.

The following example selects all elements with a class attribute value that ends with "test":

Note: The value does not have to be a whole word!

```
[class$="test"] {
  background: yellow;
}
```

[attribute*="value"] Selector

The `[attribute*="value"]` selector is used to select elements whose attribute value contains a specified value.

The following example selects all elements with a class attribute value that contains "te":

Note: The value does not have to be a whole word!

```
[class*="te"] {
  background: yellow;
}
```

Styling Forms

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

```
input[type="text"] {
  width: 150px;
  display: block;
  margin-bottom: 10px;
  background-color: yellow;
}

input[type="button"] {
  width: 120px;
  margin-left: 35px;
  display: block;
}
```

Math Functions

The CSS math functions allow mathematical expressions to be used as property values. Here, we will explain the `calc()`, `max()` and `min()` functions.

Function	Description
calc()	Allows you to perform calculations to determine CSS property values
max()	Uses the largest value, from a comma-separated list of values, as the property value
min()	Uses the smallest value, from a comma-separated list of values, as the property value

calc() Function

The `calc()` function performs a calculation to be used as the property value.

`calc(`*`expression`*`)`

Value	Description
expression	Required. A mathematical expression. The result will be used as the value. The following operators can be used: + - * /

Let us look at an example:

Use calc() to calculate the width of a element:

```
#div1 {
  position: absolute;
  left: 50px;
  width: calc(100% - 100px);
  border: 1px solid black;
  background-color: yellow;  padding: 5px;
}
```

max() Function

The `max()` function uses the largest value, from a comma-separated list of values, as the property value.

`max(value1, value2, ...)`

Value	Description
value1, *value2*, ...	Required. A list of comma-separated values - where the largest value is chosen

Let us look at an example:

Use max() to set the width of #div1 to whichever value is largest, 50% or 300px:

```
#div1 {
  background-color: yellow;
  height: 100px;
  width: max(50%, 300px);
}
```

min() Function

The `min()` function uses the smallest value, from a comma-separated list of values, as the property value.

`min(value1, value2, ...)`

Value	Description
value1, *value2*, ...	Required. A list of comma-separated values - where the smallest value is chosen

Let us look at an example:

Use min() to set the width of #div1 to whichever value is smallest, 50% or 300px:

```
#div1 {
  background-color: yellow;
  height: 100px;
  width: min(50%, 300px);
}
```

!important

The `!important` rule in CSS is used to add more importance to a property/value than normal.

In fact, if you use the `!important` rule, it will override ALL previous styling rules for that specific property on that element!

Let us look at an example

```
<!DOCTYPE html>
<html>
<head>
<style>
#myid {
  background-color: blue;
}

.myclass {
  background-color: gray;
}

p {
  background-color: red !important;
}
</style>
</head>
<body>

<p>This is some text in a paragraph.</p>

<p class="myclass">This is some text in a paragraph.</p>

<p id="myid">This is some text in a paragraph.</p>

</body>
</html>
```

Example Explained

In the example above. all three paragraphs will get a red background color, even though the ID selector and the class selector have a higher specificity. The `!important` rule overrides the `background-color` property in both cases.

Important About !important

The only way to override an `!important` rule is to include another `!important` rule on a declaration with the same (or higher) specificity in the source code - and here the problem starts! This makes the CSS code confusing and the debugging will be hard, especially if you have a large style sheet!

Here we have created a simple example. It is not very clear, when you look at the CSS source code, which color is considered most important:

```
#myid {
  background-color: blue !important;
}

.myclass {background-color: gray !important;}

p {  background-color: red !important;}
```

Tip: It is good to know about the `!important` rule. You might see it in some CSS source code. However, do not use it unless you absolutely have to.

Specificity

If there are two or more CSS rules that point to the same element, the selector with the highest specificity value will "win", and its style declaration will be applied to that HTML element.

Think of specificity as a score/rank that determines which style declaration is ultimately applied to an element.

Look at the following examples:

In this example, we have used the "p" element as selector, and specified a red color for this element. The text will be red:

```
<html>
<head>
  <style>
    p {color: red;}
  </style>
</head>
<body>

<p>Hello World!</p>

</body>
</html>
```

Now, look at example 2:

In this example, we have added a class selector (named "test"), and specified a green color for this class. The text will now be green (even though we have specified a red color for the element selector "p"). This is because the class selector is given higher priority:

```
<html> <head>  <style>
    .test {color: green;}
    p {color: red;}
  </style>
</head> <body>
<p class="test">Hello World!</p>
</body>
</html>
```

Now, look at example 3:

In this example, we have added the id selector (named "demo"). The text will now be blue, because the id selector is given higher priority:

```
<html>
<head>
  <style>
    #demo {color: blue;}
    .test {color: green;}
    p {color: red;}
  </style>
</head>
<body>

<p id="demo" class="test">Hello World!</p>

</body>
</html>
```

Now, look at example 4:

In this example, we have added an inline style for the "p" element. The text will now be pink, because the inline style is given the highest priority:

```
<html>
<head>
  <style>
    #demo {color: blue;}
    .test {color: green;}
    p {color: red;}
  </style>
</head>
<body>

<p id="demo" class="test" style="color: pink;">Hello World!</p>

</body>
</html>
```

Specificity Hierarchy

Every CSS selector has its place in the specificity hierarchy.

There are four categories which define the specificity level of a selector:

1. **Inline styles** - Example: <h1 style="color: pink;">
2. **IDs** - Example: #navbar
3. **Classes, pseudo-classes, attribute selectors** - Example: .test, :hover, [href]
4. **Elements and pseudo-elements** - Example: h1, ::before

How to Calculate Specificity?

Memorize how to calculate specificity!

Start at 0, add 100 for each ID value, add 10 for each class value (or pseudo-class or attribute selector), add 1 for each element selector or pseudo-element.

Note: Inline style gets a specificity value of 1000, and is always given the highest priority!

The table below shows some examples on how to calculate specificity values:

Selector	Specificity Value	Calculation
p	1	1
p.test	11	1 + 10
p#demo	101	1 + 100
<p style="color: pink;">	1000	1000
#demo	100	100
.test	10	10
p.test1.test2	21	1 + 10 + 10
#navbar p#demo	201	100 + 1 + 100
*	0	0 (the universal selector is ignored)

The selector with the highest specificity value will win and take effect!

Consider these three code fragments:

```
A: h1
B: h1#content
C: <h1 id="content" style="color: pink;">Heading</h1>
```

The specificity of A is 1 (one element selector)
The specificity of B is 101 (one ID reference + one element selector)
The specificity of C is 1000 (inline styling)

Since the third rule (C) has the highest specificity value (1000), this style declaration will be applied.

More Specificity Rules Examples

Equal specificity: the latest rule wins - If the same rule is written twice into the external style sheet, then the latest rule wins:

```
h1 {background-color: yellow;}
h1 {background-color: red;}
```

The universal selector (*) and inherited values have a specificity of 0 - The universal selector (*) and inherited values are ignored!

CSS Units

CSS has several different units for expressing a length.

Many CSS properties take "length" values, such as `width`, `margin`, `padding`, `font-size`, etc.

Length is a number followed by a length unit, such as `10px`, `2em`, etc.

Set different length values, using px (pixels):

```
h1 {
  font-size: 60px;
}

p {
  font-size: 25px;
  line-height: 50px;
}
```

Note: A whitespace cannot appear between the number and the unit. However, if the value is `0`, the unit can be omitted.

For some CSS properties, negative lengths are allowed.

There are two types of length units: **absolute** and **relative**.

Absolute Lengths

The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

Absolute length units are not recommended for use on screen, because screen sizes vary so much. However, they can be used if the output medium is known, such as for print layout.

Unit	Description
cm	centimetersTry it
mm	millimetersTry it
in	inches (1in = 96px = 2.54cm)Try it
px *	pixels (1px = 1/96th of 1in)Try it
pt	points (1pt = 1/72 of 1in)Try it
pc	picas (1pc = 12 pt)Try it

* Pixels (px) are relative to the viewing device. For low-dpi devices, 1px is one device pixel (dot) of the display. For printers and high resolution screens 1px implies multiple device pixels.

Relative Lengths

Relative length units specify a length relative to another length property. Relative length units scale better between different rendering mediums.

Unit	Description	
em	Relative to the font-size of the element (2em means 2 times the size of the current font)	Try it
ex	Relative to the x-height of the current font (rarely used)	Try it
ch	Relative to width of the "0" (zero)	Try it
rem	Relative to font-size of the root element	Try it
vw	Relative to 1% of the width of the viewport*	Try it
vh	Relative to 1% of the height of the viewport*	Try it
vmin	Relative to 1% of viewport's* smaller dimension	Try it
vmax	Relative to 1% of viewport's* larger dimension	Try it
%	Relative to the parent element	Try it

Tip: The em and rem units are practical in creating perfectly scalable layout!
* Viewport = the browser window size. If the viewport is 50cm wide, 1vw = 0.5cm.

Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element.

See more . W3school

Flexbox

Before the Flexbox Layout module, there were four layout modes:

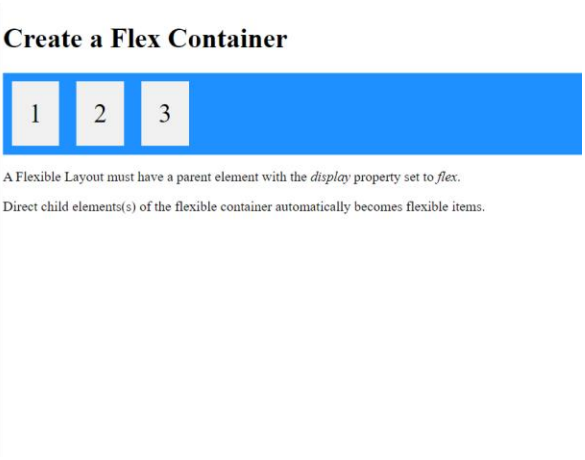
- Block, for sections in a webpage
- Inline, for text
- Table, for two-dimensional table data
- Positioned, for explicit position of an element

The Flexible Box Layout Module, makes it easier to design flexible responsive layout structure without using float or positioning.

A flex container with three flex items:

```
<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>
```

```
<!DOCTYPE html>
<html>
<head>
<style>
.flex-container {
  display: flex;
  background-color: DodgerBlue;
}

.flex-container > div {
  background-color: #f1f1f1;
  margin: 10px;
  padding: 20px;
  font-size: 30px;
}
</style>
</head>
<body>

<h1>Create a Flex Container</h1>

<div class="flex-container">
  <div>1</div>
  <div>2</div>
  <div>3</div>
</div>

<p>A Flexible Layout must have a parent element with the <em>display</em> property set
to <em>flex</em>.</p>
```



You will learn more about flex containers and flex items in the next chapters.

Flexbox Container Properties

The following table lists all the CSS Flexbox Container properties:

Property	Description
align-content	Modifies the behavior of the flex-wrap property. It is similar to align-items, but instead of aligning flex items, it aligns flex lines
align-items	Vertically aligns the flex items when the items do not use all available space on the cross-axis
display	Specifies the type of box used for an HTML element
flex-direction	Specifies the direction of the flexible items inside a flex container
flex-flow	A shorthand property for flex-direction and flex-wrap
flex-wrap	Specifies whether the flex items should wrap or not, if there is not enough room for them on one flex line
justify-content	Horizontally aligns the flex items when the items do not use all available space on the main-axis

align-content

The `align-content` property specifies how flex lines are distributed along the cross axis in a flexbox container.

In flexbox layout, the main axis is in the `flex-direction` (default is 'row', horizontal), and the cross axis is perpendicular to the main axis (default is 'column', vertical).

CSS Syntax

```
align-content: stretch|center|flex-start|flex-end|space-between|space-around|space-evenly|initial|inherit;
```

Value	Description	Demo
stretch	Default value. Lines stretch to take up the remaining space	Demo ❯
center	Lines are packed toward the center of the flex container	Demo ❯
flex-start	Lines are packed toward the start of the flex container	Demo ❯
flex-end	Lines are packed toward the end of the flex container	Demo ❯
space-between	Lines are evenly distributed in the flex container	Demo ❯
space-around	Lines are evenly distributed in the flex container, with half-size spaces on either end	Demo ❯
space-evenly	Lines are evenly distributed in the flex container, with equal space around them	Demo ❯
initial	Sets this property to its default value. Read about *initial*	
inherit	Inherits this property from its parent element. Read about *inherit*	

display

CSS Syntax

```
display: value;
```

Property Values

Value	Description
inline	Displays an element as an inline element (like). Any height and width properties
block	Displays an element as a block element (like <p>). It starts on a new line, and takes up th
contents	Makes the container disappear, making the child elements children of the element the n
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 apply height and width values
inline-flex	Displays an element as an inline-level flex container
inline-grid	Displays an element as an inline-level grid container
inline-table	The element is displayed as an inline-level table
list-item	Let the element behave like a element
run-in	Displays an element as either block or inline, depending on context
table	Let the element behave like a <table> element
table-caption	Let the element behave like a <caption> element
table-column-group	Let the element behave like a <colgroup> element
table-header-group	Let the element behave like a <thead> element
table-footer-group	Let the element behave like a <tfoot> element
table-row-group	Let the element behave like a <tbody> element
table-cell	Let the element behave like a <td> element
table-column	Let the element behave like a <col> element
table-row	Let the element behave like a <tr> element
none	The element is completely removed
initial	Sets this property to its default value. Read about *initial*
inherit	Inherits this property from its parent element. Read about *inherit*

flex-direction

```
flex-direction: row|row-reverse|column|column-reverse|initial|inherit;
```

Property Values

Value	Description	Play it
row	Default value. The flexible items are displayed horizontally, as a row	Demo ›
row-reverse	Same as row, but in reverse order	Demo ›
column	The flexible items are displayed vertically, as a column	Demo ›
column-reverse	Same as column, but in reverse order	Demo ›
initial	Sets this property to its default value. Read about *initial*	
inherit	Inherits this property from its parent element. Read about *inherit*	

flex-wrap

```
flex-wrap: nowrap|wrap|wrap-reverse|initial|inherit;
```

Property Values

Value	Description	Play it
nowrap	Default value. Specifies that the flexible items will not wrap	Demo ›
wrap	Specifies that the flexible items will wrap if necessary	Demo ›
wrap-reverse	Specifies that the flexible items will wrap, if necessary, in reverse order	Demo ›
initial	Sets this property to its default value. Read about *initial*	
inherit	Inherits this property from its parent element. Read about *inherit*	

justify-content

```
justify-content: flex-start|flex-end|center|space-between|space-around|space-evenly|initial|inherit;
```

Property Values

Value	Description	Play it
flex-start	Default value. Items are positioned at the beginning of the container	Demo ❯
flex-end	Items are positioned at the end of the container	Demo ❯
center	Items are positioned in the center of the container	Demo ❯
space-between	Items will have space between them	Demo ❯
space-around	Items will have space before, between, and after them	Demo ❯
space-evenly	Items will have equal space around them	Demo ❯
initial	Sets this property to its default value. Read about *initial*	
inherit	Inherits this property from its parent element. Read about *inherit*	

Transitions

CSS transitions allows you to change property values smoothly, over a given duration.

- `transition`
- `transition-delay`
- `transition-duration`
- `transition-property`
- `transition-timing-function`

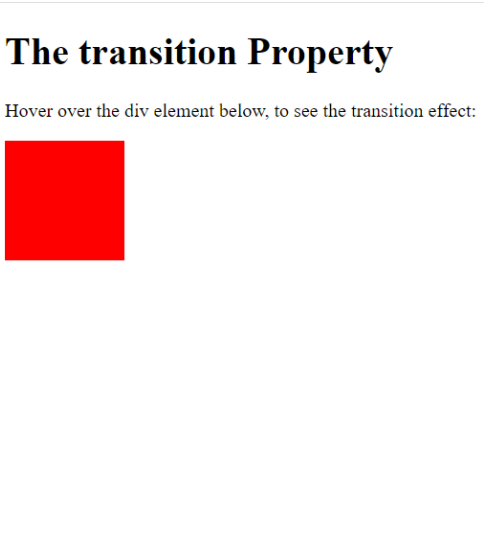
How to Use CSS Transitions?

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect

Note: If the duration part is not specified, the transition will have no effect, because the default value is 0.

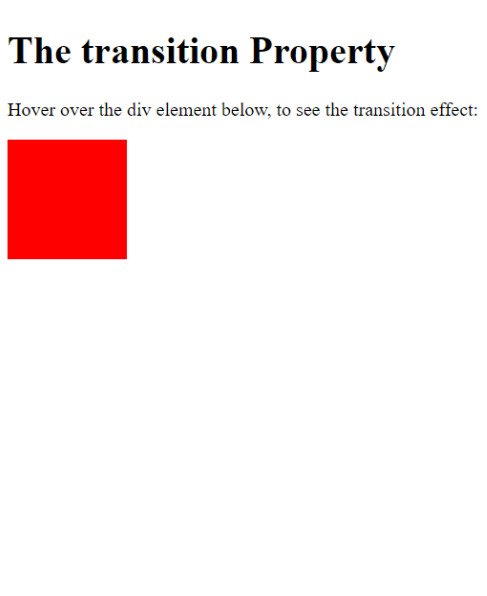
```html
<!DOCTYPE html>
<html>
<head>
<style>
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}

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

<h1>The transition Property</h1>

<p>Hover over the div element below, to see the transition effect:</p>
<div></div>

</body>
</html>
```



Change Several Property Values

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

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

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

<h1>The transition Property</h1>

<p>Hover over the div element below, to see the transition effect:</p>

<div></div>

</body>
</html>
```



Specify the Speed Curve of the Transition

The `transition-timing-function` property specifies the speed curve of the transition effect.

The transition-timing-function property can have the following values:

- `ease` - specifies a transition effect with a slow start, then fast, then end slowly (this is default)
- `linear` - specifies a transition effect with the same speed from start to end
- `ease-in` - specifies a transition effect with a slow start
- `ease-out` - specifies a transition effect with a slow end
- `ease-in-out` - specifies a transition effect with a slow start and end
- `cubic-bezier(n,n,n,n)` - lets you define your own values in a cubic-bezier function

The following example shows some of the different speed curves that can be used:

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

Delay the Transition Effect

The `transition-delay` property specifies a delay (in seconds) for the transition effect.

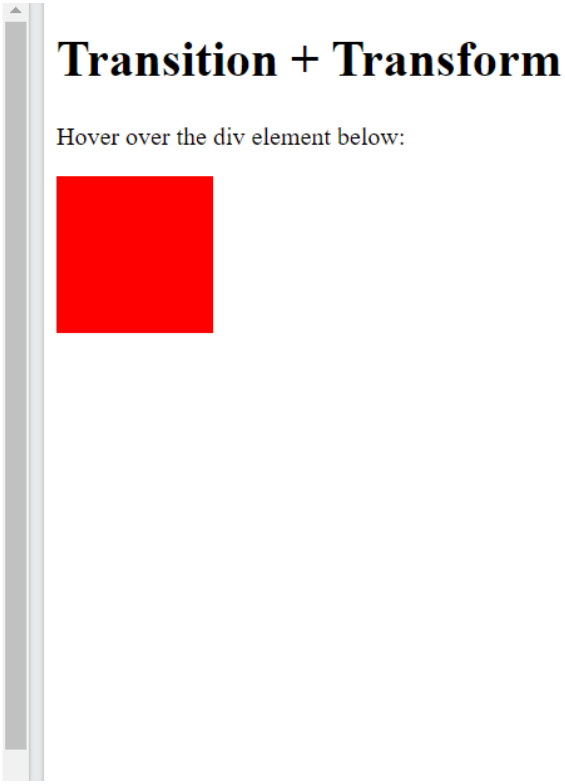
The following example has a 1 second delay before starting:

```
div {
  transition-delay: 1s;
}
```

Transition + Transformation

The following example adds a transition effect to the transformation:

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

div:hover {
  width: 300px;
  height: 300px;
  transform: rotate(180deg);
}
</style>
</head>
<body>

<h1>Transition + Transform</h1>

<p>Hover over the div element below:</p>

<div></div>

</body>
</html>
```



CSS Transition Properties

The following table lists all the CSS transition properties:

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

CSS gradients

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines three types of gradients:

- **Linear Gradients (goes down/up/left/right/diagonally)**
- **Radial Gradients (defined by their center)**
- **Conic Gradients (rotated around a center point)**

Linear Gradients

To create a linear gradient you must define at least two color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax

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

Radial Gradients

A radial gradient is defined by its center.

To create a radial gradient you must also define at least two color stops.

Syntax

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

By default, shape is ellipse, size is farthest-corner, and position is center.

Conic Gradients

A conic gradient is a gradient with color transitions rotated around a center point.

To create a conic gradient you must define at least two colors.

Syntax

```
background-image: conic-gradient([from angle]
[at position,] color [degree], color [degree], ...);
```

By default, *angle* is 0deg and *position* is center.

If no *degree* is specified, the colors will be spread equally around the center point.

CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

Mouse over the element below to see a 2D transformation:

The following table lists all the 2D transform properties:

Property	Description
transform	Applies a 2D or 3D transformation to an element
transform-origin	Allows you to change the position on transformed elements

CSS 2D Transform Methods

Function	Description
matrix(*n,n,n,n,n,n*)	Defines a 2D transformation, using a matrix of six values
translate(*x,y*)	Defines a 2D translation, moving the element along the X- and the Y-axis
translateX(*n*)	Defines a 2D translation, moving the element along the X-axis
translateY(*n*)	Defines a 2D translation, moving the element along the Y-axis
scale(*x,y*)	Defines a 2D scale transformation, changing the elements width and height
scaleX(*n*)	Defines a 2D scale transformation, changing the element's width
scaleY(*n*)	Defines a 2D scale transformation, changing the element's height
rotate(*angle*)	Defines a 2D rotation, the angle is specified in the parameter
skew(*x-angle,y-angle*)	Defines a 2D skew transformation along the X- and the Y-axis
skewX(*angle*)	Defines a 2D skew transformation along the X-axis
skewY(*angle*)	Defines a 2D skew transformation along the Y-axis

```
div {
  transform: translate(50px, 100px);
}
```

```
div {
  transform: rotate(20deg);
}
```

CSS 3D Transforms

CSS also supports 3D transformations.

With the CSS `transform` property you can use the following 3D transformation methods:

- `rotateX()`
- `rotateY()`
- `rotateZ()`

```
#myDiv {
  transform: rotateX(150deg);
}
```

```
#myDiv {
  transform: rotateY(150deg);
}
```

```
#myDiv {
  transform: rotateZ(90deg);
}
```

User Interface

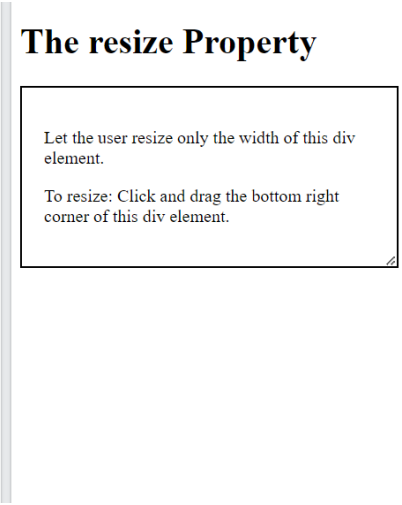
In this chapter you will learn about the following CSS user interface properties:

- resize
- outline-offset

Resizing

The resize property specifies if (and how) an element should be resizable by the user.

```
<!DOCTYPE html>
<html>
<head>
<style>
div {
  border: 2px solid;
  padding: 20px;
  width: 300px;
  resize: horizontal;
  overflow: auto;
}
</style>
</head>
<body>

<h1>The resize Property</h1>

<div>
  <p>Let the user resize only the width of this div element.</p>
  <p>To resize: Click and drag the bottom right corner of this div element.</p>
</div>

</body>
</html>
```



The following example lets the user resize only the width of a element:

```
div {
  resize: horizontal;
  overflow: auto;
}
```

The following example lets the user resize only the height of a element:

```
div {
  resize: vertical;
  overflow: auto;
}
```

The following example lets the user resize both the height and width of a element:

```
div {
  resize: both;
  overflow: auto;
}
```

In many browsers, <textarea> is resizable by default. Here, we have used the resize property to disable the resizability:

```
textarea {
  resize: none;
}
```

Outline Offset

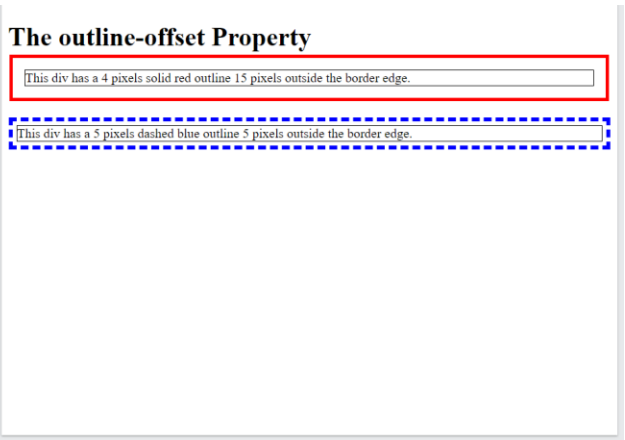
The `outline-offset` property adds space between an outline and the edge or border of an element.

This div has an outline 15px outside the border edge.

Note: Outline differs from borders! Unlike border, the outline is drawn outside the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

The following example uses the `outline-offset` property to add space between the border and the outline:

```html
<!DOCTYPE html>
<html>
<head>
<style>
div.ex1 {
  margin: 20px;
  border: 1px solid black;
  outline: 4px solid red;
  outline-offset: 15px;
}

div.ex2 {
  margin: 10px;
  border: 1px solid black;
  outline: 5px dashed blue;
  outline-offset: 5px;
}
</style>
</head>
<body>
<h1>The outline-offset Property</h1>

<div class="ex1">This div has a 4 pixels solid red outline 15 pixels outside the border edge.</div>
<br>

<div class="ex2">This div has a 5 pixels dashed blue outline 5 pixels outside the border edge.</div>

</body>
```



Variables

The `var()` function is used to insert the value of a CSS variable.

CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavaScript, and change the variables based on media queries.

A good way to use CSS variables is when it comes to the colors of your design. Instead of copy and paste the same colors over and over again, you can place them in variables.

Syntax of the var() Function

The `var()` function is used to insert the value of a CSS variable.

The syntax of the `var()` function is as follows:

```
var(--name, value)
```

Value	Description
name	Required. The variable name (must start with two dashes)
value	Optional. The fallback value (used if the variable is not found)

Note: The variable name must begin with two dashes (--) and it is case sensitive!

How var() Works

First of all: CSS variables can have a global or local scope.

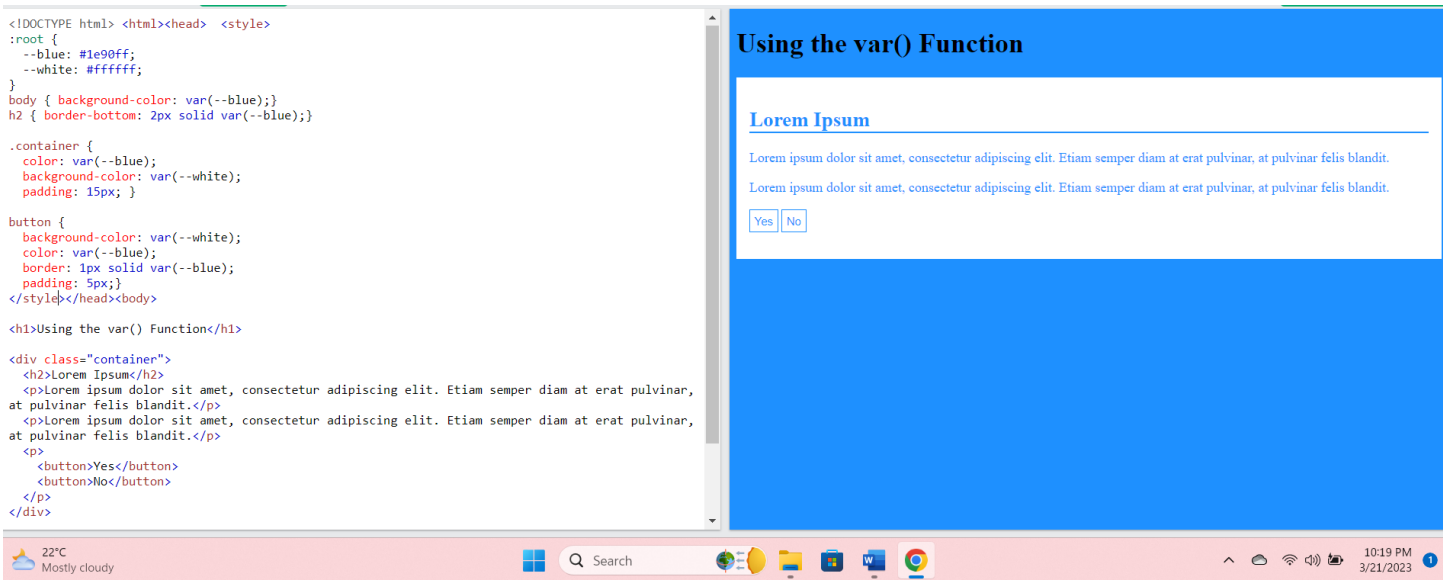
Global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.

To create a variable with global scope, declare it inside the `:root` selector. The `:root` selector matches the document's root element.

To create a variable with local scope, declare it inside the selector that is going to use it.

The following example is equal to the example above, but here we use the `var()` function.

First, we declare two global variables (--blue and --white). Then, we use the `var()` function to insert the value of the variables later in the style sheet:



```
<!DOCTYPE html> <html><head>  <style>
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}
body { background-color: var(--blue);}
h2 { border-bottom: 2px solid var(--blue);}

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px; }

button {
  background-color: var(--white);
  color: var(--blue);
  border: 1px solid var(--blue);
  padding: 5px;}
</style></head><body>

<h1>Using the var() Function</h1>

<div class="container">
  <h2>Lorem Ipsum</h2>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar,
at pulvinar felis blandit.</p>
  <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar,
at pulvinar felis blandit.</p>
  <p>
    <button>Yes</button>
    <button>No</button>
  </p>
</div>
```

Using the var() Function

Lorem Ipsum

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam semper diam at erat pulvinar, at pulvinar felis blandit.

Yes No

22°C Mostly cloudy

Search

10:19 PM 3/21/2023

Advantages of using var() are:

- makes the code easier to read (more understandable)
- makes it much easier to change the color values

Override Global Variable With Local Variable

From the previous page we have learned that global variables can be accessed/used through the entire document, while local variables can be used only inside the selector where it is declared.

Look at the example from the previous page:

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}

body {
  background-color: var(--blue);
}

h2 {
  border-bottom: 2px solid var(--blue);
}

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
}

button {
  background-color: var(--white);
  color: var(--blue);
  border: 1px solid var(--blue);
  padding: 5px;
}
```

Sometimes we want the variables to change only in a specific section of the page.

Assume we want a different color of blue for button elements. Then, we can re-declare the --blue variable inside the button selector. When we use var(--blue) inside this selector, it will use the local --blue variable value declared here.

We see that the local --blue variable will override the global --blue variable for the button elements:

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}

body {
  background-color: var(--blue);
}

h2 {
  border-bottom: 2px solid var(--blue);
}

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
}

button {
  --blue: #0000ff; /* local variable will override global */
  background-color: var(--white);
  color: var(--blue);
  border: 1px solid var(--blue);
  padding: 5px;
```

Add a New Local Variable

If a variable is to be used at only one single place, we could also have declared a new local variable, like this:

```
:root {
  --blue: #1e90ff;
  --white: #ffffff;
}

body {
  background-color: var(--blue);
}

h2 {
  border-bottom: 2px solid var(--blue);
}

.container {
  color: var(--blue);
  background-color: var(--white);
  padding: 15px;
}

button {
  --button-blue: #0000ff; /* new local variable */
  background-color: var(--white);
  color: var(--button-blue);
  border: 1px solid var(--button-blue);
  padding: 5px;
}
```

Box Sizing

The CSS `box-sizing` property allows us to include the padding and border in an element's total width and height.

Without the CSS box-sizing Property

By default, the width and height of an element is calculated like this:

width + padding + border = actual width of an element
height + padding + border = actual height of an element

This means: When you set the width/height of an element, the element often appears bigger than you have set (because the element's border and padding are added to the element's specified width/height).

The following illustration shows two elements with the same specified width and height:

This div is smaller (width is 300px and height is 100px).

This div is bigger (width is also 300px and height is 100px).

```
<!DOCTYPE html>
<html>
<head>
<style>
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
}

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
}
</style>
</head>
<body>

<h1>Without box-sizing</h1>

<div class="div1">This div is smaller (width is 300px and height is 100px).</div>
<br>
<div class="div2">This div is bigger (width is also 300px and height is 100px).</div>

</body>
</html>
```

Without box-sizing

This div is smaller (width is 300px and height is 100px).

This div is bigger (width is also 300px and height is 100px).

The box-sizing property solves this problem

With the CSS box-sizing Property

The box-sizing property allows us to include the padding and border in an element's total width and height.

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

Both divs are the same size now!

Hooray!

```
<!DOCTYPE html>
<html>
<head>
<style>
.div1 {
  width: 300px;
  height: 100px;
  border: 1px solid blue;
  box-sizing: border-box;
}

.div2 {
  width: 300px;
  height: 100px;
  padding: 50px;
  border: 1px solid red;
  box-sizing: border-box;
}
</style>
</head>
<body>

<h1>With box-sizing</h1>

<div class="div1">Both divs are the same size now!</div>
<br>
<div class="div2">Hooray!</div>

</body>
</html>
```

With box-sizing

Both divs are the same size now!

Hooray!

Media Queries

The `@media` rule, introduced in CSS2, made it possible to define different style rules for different media types.

Examples: You could have one set of style rules for computer screens, one for printers, one for handheld devices, one for television-type devices, and so on.

Unfortunately these media types never got a lot of support by devices, other than the print media type.

CSS3 Introduced Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.

Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

Media Query Syntax

A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

```
@media not|only mediatype and (expressions) {
  CSS-Code;
}
```

The result of the query is true if the specified media type matches the type of device the document is being displayed on and all expressions in the media query are true. When a media query is true, the corresponding style sheet or style rules are applied, following the normal cascading rules.

Unless you use the not or only operators, the media type is optional and the `all` type will be implied.

You can also have different stylesheets for different media:

```
<link rel="stylesheet" media="mediatype and|not|only
(expressions)" href="print.css">
```

CSS3 Media Types

Value	Description
all	Used for all media type devices
print	Used for printers
screen	Used for computer screens, tablets, smart-phones etc.
speech	Used for screenreaders that "reads" the page out loud

Media Queries Simple Examples

One way to use media queries is to have an alternate CSS section right inside your style sheet.

The following example changes the background-color to lightgreen if the viewport is 480 pixels wide or wider (if the viewport is less than 480 pixels, the background-color will be pink):