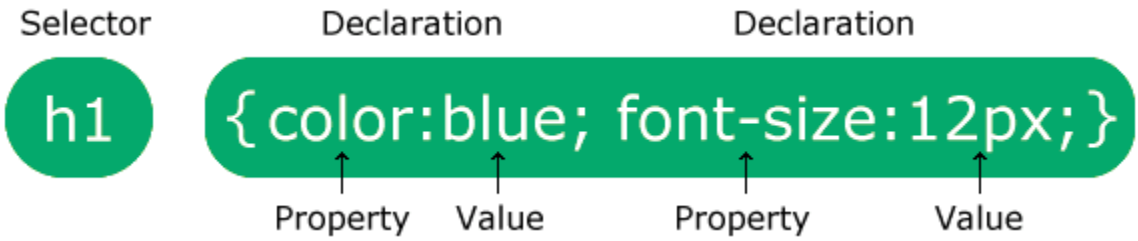
## What is CSS?

- 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.

## CSS Syntax




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

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

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

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

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

Example Explained

- p is a selector in CSS (it points to the HTML element you want to style: <p>).
- color is a property, and red is the property value

- text-align is a property, and center is the property value

## CSS 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)

## The CSS element Selector

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

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

The CSS 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.

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

## The CSS 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.

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

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

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

HTML elements can also refer to more than one class.

## The CSS Universal Selector

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

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

## The CSS 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.

### Three Ways to Insert CSS

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.

```
<!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.

## 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.

```
<!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.

```
<!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 stylesheet will be used.

## CSS Comments

Comments are used to explain the code, and may help when you edit the source code at a later date.

Comments are ignored by browsers.

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

```
/* This is a single-line comment */
p {
  color: red;
}
```

## CSS Backgrounds

The CSS background properties are used to add background effects for elements.

In these chapters, you will learn about the following CSS background properties:

- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background (shorthand property)

## CSS background-color

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

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

```
}
```

## 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:

```
div {
  background-color: green;
  opacity: 0.3;
}
```

## CSS background-image

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

By default, the image is repeated so it covers the entire element.

```
body {
  background-image: url("paper.gif");
}
```

The background image can also be set for specific elements, like the <p> element:

```
p {
  background-image: url("paper.gif");
}
```

## CSS 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:

```
body {
  background-image: url("gradient_bg.png");
}
```

## CSS background-repeat: no-repeat

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

```
body {
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
}
```

## CSS background-position

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

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

## CSS background-attachment

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

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

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

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

## CSS 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:

```
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.

## CSS Borders

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

## CSS Border Style

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

The following values are allowed:

- dotted - Defines a dotted border

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

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

```
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;}
```

## CSS Border Width

The border-width property specifies the width of the four borders.

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

```
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;
}
```

## 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):

```
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 */
}
```

## CSS Border Color

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

The color can be set by:

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

```
p.one {
```

```
  border-style: solid;
  border-color: red;
}
```

## 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).

```
p.one {
  border-style: solid;
  border-color: red green blue yellow; /* red top, green right, blue bottom and yellow left */
}
```

## CSS Border - Individual Sides

From the examples on the previous pages, you have seen that it is possible to specify a different border for each side.

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;
}
```

The example above gives the same result as this:

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

## CSS Border - Shorthand Property

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;
}
```

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

```
p {
  border-left: 6px solid red;
}
```

## CSS Rounded Borders

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

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

## CSS Margins

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

## CSS 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

```
p {
  margin-top: 100px;
  margin-bottom: 100px;
  margin-right: 150px;
  margin-left: 80px;
}
```

## Margin - Shorthand Property

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

The margin property is a shorthand property for the following individual margin properties:

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

So, here is how it works:

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

```
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

```
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

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

If the margin property has one value:

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

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

## The auto Value

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.

```
div {
  width: 300px;
  margin: auto;
```

```
  border: 1px solid red;
}
```

## The inherit Value

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

```
div {
  border: 1px solid red;
  margin-left: 100px;
}

p.ex1 {
  margin-left: inherit;
}
```

## 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!

Look at the following example:

```
h1 {
  margin: 0 0 50px 0;
}

h2 {
  margin: 20px 0 0 0;
}
```

## CSS 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.

```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

## 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

```
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

```
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

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

If the padding property has one value:

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

```
div {
```

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

## CSS 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.

## CSS Setting height and width

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

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

## 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

```
div {
  height: 200px;
  width: 50%;
  background-color: powderblue;
}
```

## 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  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.

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).

```
div {
  max-width: 500px;
  height: 100px;
  background-color: powderblue;
}
```

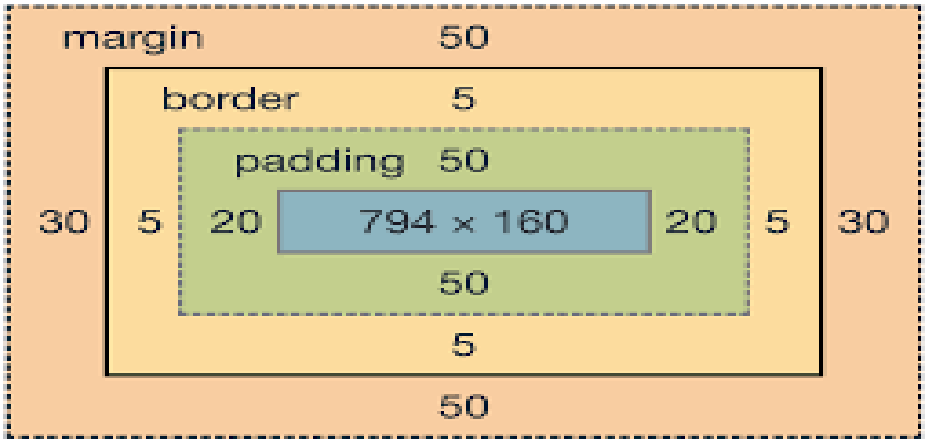
## CSS Box Model

All HTML elements can be considered as boxes.

## The CSS 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

The box model allows us to add a border around elements, and to define space between elements.

```
div {
  width: 300px;
  border: 15px solid green;
  padding: 50px;
  margin: 20px;
}
```

## CSS Outline

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

```
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;}
```

## 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)

```
p.ex1 {

  border: 1px solid black;

  outline-style: solid;

  outline-color: red;

  outline-width: thin;

}
```

```
p.ex2 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: medium;
}

p.ex3 {
  border: 1px solid black;
  outline-style: solid;
  outline-color: red;
  outline-width: thick;
}
```

```
p.ex4 {

  border: 1px solid black;

  outline-style: solid;

  outline-color: red;

  outline-width: 4px;

    }
```

## CSS Outline Color

The outline-color property is used to set the color of the outline.

```
p.ex1 {
  border: 2px solid black;
  outline-style: solid;
  outline-color: red;
}

p.ex2 {
  border: 2px solid black;
  outline-style: dotted;
  outline-color: blue;
}

p.ex3 {
  border: 2px solid black;
  outline-style: outset;
  outline-color: grey;
}
```

## 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

The outline property is specified as one, two, or three values from the list above. The order of the values does not matter.

```
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:

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

## Text Decoration

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

## Add a Decoration Line to Text

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

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

```
h1 {
  text-decoration-line: overline;
}

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

h3 {
  text-decoration-line: underline;
}

p {
  text-decoration-line: overline underline;
}
```

## Specify a Color for the Decoration Line

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

```
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;
}
```

## Specify a Style for the Decoration Line

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

```
h1 {
  text-decoration-line: underline;
  text-decoration-style: solid;
}

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;
}
```

## Specify the Thickness for the Decoration Line

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

```
h1 {
  text-decoration-line: underline;
  text-decoration-thickness: auto;
}

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;
}
```

## 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)

```
h1 {
  text-decoration: underline;
}

h2 {
  text-decoration: underline red;
}
```

```
h3 {
  text-decoration: underline red double;
}

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

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;
}
```

## 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:

```
p.uppercase {
  text-transform: uppercase;
}

p.lowercase {
  text-transform: lowercase;
}

p.capitalize {
  text-transform: capitalize;
}
```

## Text Spacing

In this chapter you will learn about the following properties:

- 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:

```
p {
  text-indent: 50px;
}
```

## Letter Spacing

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

The following example demonstrates how to increase or decrease the space between characters:

```
h1 {
  letter-spacing: 5px;
}

h2 {
  letter-spacing: -2px;
}
```

## Line Height

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

```
p.small {
  line-height: 0.8;
}

p.big {
  line-height: 1.8;
}
```

## Word Spacing

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

The following example demonstrates how to increase or decrease the space between words:

```
p.one {
  word-spacing: 10px;
}

p.two {
  word-spacing: -2px;
}
```

## White Space

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

This example demonstrates how to disable text wrapping inside an element:

```
p {
  white-space: nowrap;
}
```

## Text Shadow

The text-shadow property adds shadow to text.

In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Text shadow effect!

```
h1 {
  text-shadow: 2px 2px;
}
```

Next, add a color (red) to the shadow:

Text shadow effect!

```
h1 {
  text-shadow: 2px 2px red;
}
```

Then, add a blur effect (5px) to the shadow:

Text shadow effect!

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

# Styling Links

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

```
a {
  color: hotpink;
}
```

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

# Text Decoration

The text-decoration property is mostly used to remove underlines from links:

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

a:visited {
  text-decoration: none;
}

a:hover {
  text-decoration: underline;
}
```

```
a:active {
  text-decoration: underline;
}
```

# Background Color

The background-color property can be used to specify a background color for links:

```
a:link {
  background-color: yellow;
}

a:visited {
  background-color: cyan;
}

a:hover {
  background-color: lightgreen;
}

a:active {
  background-color: hotpink;
}
```

# 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.

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

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.

```
span {
  display: block;
}
```

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:

```
h1.hidden {
  display: none;
}
```

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:

# 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:

```
div.static {

  position: static;

  border: 3px solid #73AD21;

}
```

# 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.

```
div.relative {

  position: relative;

  left: 30px;

  border: 3px solid #73AD21;

}
```

# 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:

```
div.fixed {
```

```
  position: fixed;
  bottom: 0;
  right: 0;
  width: 300px;
  border: 3px solid #73AD21;
}
```

# 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.

```
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;

}
```

# 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).

Note:  Internet Explorer does not support sticky positioning. Safari requires a -webkit- prefix (see example below). You must also specify at least one of top, right, bottom or left for sticky positioning to work.

```
div.sticky {
  position: -webkit-sticky; /* Safari */
  position: sticky;
  top: 0;
  background-color: green;
  border: 2px solid #4CAF50;
}
```

# 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:

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

# CSS 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:

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 {
  width: 200px;
  height: 65px;
  background-color: coral;
  overflow: visible;
}
```

# overflow: hidden

With the hidden value, the overflow is clipped, and the rest of the content is hidden:

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what

```
div {
  overflow: hidden;
```

```
}
```

# 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):

```
div {
  overflow: scroll;
}
```

# overflow: auto

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

```
div {
  overflow: auto;
}
```

# 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 */
}
```

# CSS 2D Transforms

CSS transforms allow you to move, rotate, scale, and skew elements.

# CSS 2D Transforms Methods

With the CSS transform property you can use the following 2D transformation methods:

- translate()
- rotate()
- scaleX()
- scaleY()
- scale()
- skewX()
- skewY()
- skew()
- matrix()

# The translate() Method

The translate() method moves an element from its current position (according to the parameters given for the X-axis and the Y-axis).

The following example moves the  element 50 pixels to the right, and 100 pixels down from its current position:

```
div {
  transform: translate(50px, 100px);
}
```

# The rotate() Method

The rotate() method rotates an element clockwise or counter-clockwise according to a given degree.

The following example rotates the  element clockwise with 20 degrees:

```
div {
  transform: rotate(20deg);
}
```

Using negative values will rotate the element counter-clockwise.

The following example rotates the  element counter-clockwise with 20 degrees:

```
div {
  transform: rotate(-20deg);
}
```

# The scale() Method

The scale() method increases or decreases the size of an element (according to the parameters given for the width and height).

The following example increases the  element to be two times of its original width, and three times of its original height:

```
div {
  transform: scale(2, 3);
}
```

The following example decreases the  element to be half of its original width and height:

```
div {
  transform: scale(0.5, 0.5);
}
```

# The scaleX() Method

The scaleX() method increases or decreases the width of an element.

The following example increases the  element to be two times of its original width:

```
div {
  transform: scaleX(2);
}
```

The following example decreases the  element to be half of its original width:

```
div {
  transform: scaleX(0.5);
}
```

# The scaleY() Method

The scaleY() method increases or decreases the height of an element.

The following example increases the  element to be three times of its original height:

```
div {
  transform: scaleY(3);
}
```

The following example decreases the  element to be half of its original height:

```
div {
  transform: scaleY(0.5);
}
```

# The skewX() Method

The skewX() method skews an element along the X-axis by the given angle.

The following example skews the  element 20 degrees along the X-axis:

```
div {
  transform: skewX(20deg);
}
```

# The skewY() Method

The skewY() method skews an element along the Y-axis by the given angle.

The following example skews the  element 20 degrees along the Y-axis:

```
div {
  transform: skewY(20deg);
}
```

# The skew() Method

The skew() method skews an element along the X and Y-axis by the given angles.

The following example skews the  element 20 degrees along the X-axis, and 10 degrees along the Y-axis:

```
div {
  transform: skew(20deg, 10deg);
}
```

If the second parameter is not specified, it has a zero value. So, the following example skews the  element 20 degrees along the X-axis:

```
div {
  transform: skew(20deg);
}
```

# The matrix() Method

The matrix() method combines all the 2D transform methods into one.

The matrix() method takes six parameters, containing mathematical functions, which allows you to rotate, scale, move (translate), and skew elements.

The parameters are as follow: matrix(scaleX(), skewY(), skewX(), scaleY(), translateX(), translateY())

```
div {
  transform: matrix(1, -0.3, 0, 1, 0, 0);
}
```

# CSS 3D Transforms

CSS also supports 3D transformations.

# CSS 3D Transforms Methods

With the CSS transform property you can use the following 3D transformation methods:

- rotateX()
- rotateY()
- rotateZ()

# The rotateX() Method

The rotateX() method rotates an element around its X-axis at a given degree:

```
#myDiv {
  transform: rotateX(150deg);
}
```

# The rotateY() Method

The rotateY() method rotates an element around its Y-axis at a given degree:

```
#myDiv {
  transform: rotateY(150deg);
}
```

# The rotateZ() Method

The rotateZ() method rotates an element around its Z-axis at a given degree:

```
#myDiv {
  transform: rotateZ(90deg);
}
```