CSS

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.

HTML was NEVER intended to contain tags for formatting a web page!

HTML was created to describe the content of a web page, like:

<h1>This is a heading</h1>

This is a paragraph.

When tags like <font>, and color attributes were added to the HTML 3.2 specification, it started a nightmare for web developers. Development of large websites, where fonts and color information were added to every single page, became a long and expensive process.

To solve this problem, the World Wide Web Consortium (W3C) created CSS.

CSS removed the style formatting from the HTML page!

CSS Saves a Lot of Work!

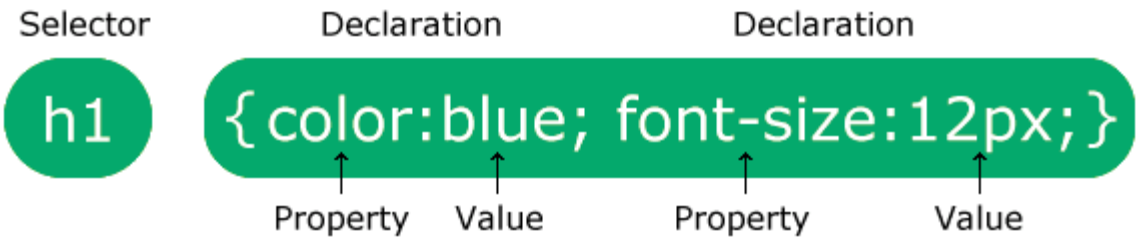
The style definitions are normally saved in external .css files.

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

CSS Syntax

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

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.

Example

In this example all <p> elements will be center-aligned, with a red text color:

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

- `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

A CSS selector selects the HTML element(s) you want to style.

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)

This page will explain the most basic CSS selectors.

The CSS element Selector

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

Example

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

```
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.

Example

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

```
#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.

Example

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

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

Example

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

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

HTML elements can also refer to more than one class.

Example

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>
```

The CSS Universal Selector

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

Example

The CSS rule below will affect every HTML element 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.

Example

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

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

How To Add CSS

When a browser reads a style sheet, it will format the HTML document according to the information in the style sheet.

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.

Example

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.

Example

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.

Example

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>
```

CSS Comments

CSS comments are not displayed in the browser, but they can help document your source code. 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 `*/`:

Example

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

You can add comments wherever you want in the code:

Example

```
p {
  color: red;  /* Set text color to red */
}
```

Example

```
p {
  color: /*red*/blue;
}
```

Comments can also span multiple lines:

Example

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

p {
  color: red;
}
```

CSS Colors

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

CSS Color Names

In CSS, a color can be specified by using a predefined color name:

Example

```
<h1 style="background-color:DodgerBlue;">Hello World</h1>
<p style="background-color:Tomato;">Lorem ipsum...</p>
```

CSS Text Color

You can set the color of text:

Example

```
<h1 style="color:Tomato;">Hello World</h1>
<p style="color:DodgerBlue;">Lorem ipsum...</p>
<p style="color:MediumSeaGreen;">Ut wisi enim...</p>
```

CSS Border Color

You can set the color of borders:

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>
```

CSS RGB Colors

An RGB color value represents RED, GREEN, and BLUE light sources.

RGB Value

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).

RGBA Value

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):

CSS HEX Colors

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

HEX Value

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.

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.

Here is an example:

Example

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

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

p {
```

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

CSS HSL Colors

HSL stands for hue, saturation, and lightness.

HSL Value

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

HSLA Value

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color.

An HSLA color value is specified with:

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

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

CSS Backgrounds

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

- `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.

Example

The background color of a page is set like this:

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

CSS background-image

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

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

Example

Set the background image for a page:

```
body {
  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:

Example

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

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

Example

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

CSS background-repeat: no-repeat

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

Example

Show the background image only once:

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

In the example above, the background image is placed in the same place as the text. We want to change the position of the image, so that it does not disturb the text too much.

CSS background-position

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

Example

Position the background image in the top-right corner:

```
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):

Example

Specify that the background image should be fixed:

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

Example

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`:

Example

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

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

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).

Example

Demonstration of the different border styles:

```
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:

Example

Demonstration of the different border widths:

```
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):

Example

```
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:

- 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.

Example

Demonstration of the different border colors:

```
p.one {
  border-style: solid;
  border-color: red;
}

p.two {
  border-style: solid;
```

```
  border-color: green;
}

p.three {
  border-style: dotted;
  border-color: blue;
}
```

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):

Example

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

CSS 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`

Example

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

CSS 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`

Example

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

CSS Rounded Borders

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

Example

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

CSS Margins

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

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.

Example

Set different margins for all four sides of a <p> 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

Example

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

Example

Use the margin shorthand property with three values:

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

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.

Example

Use margin: auto:

```css
div {
  width: 300px;
  margin: auto;
  border: 1px solid red;
}
```

The inherit Value

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

Example

Use of the inherit value:

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

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

Margin Collapse

Sometimes two margins collapse into a single margin.

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:

Example

Demonstration of margin collapse:

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

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

CSS Padding

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

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.

Example

Set different padding for all four sides of a element:

```
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

Example

Use the padding shorthand property with four values:

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

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.

Example

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;
}
```

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.

Example

Use the box-sizing property to keep the width at 300px, no matter the amount of padding:

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

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

Example

Set the height and width of a element:

```
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 `<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).

Example

This element has a height of 100 pixels and a max-width of 500 pixels:

```css
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: content, padding, borders and margins. 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.

Example

Demonstration of the box model:

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

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 total width and height of an element, you must also include the padding and borders.

Example

This element will have a total width of 350px and a total height of 80px:

```
div {
  width: 320px;
  height: 50px;
  padding: 10px;
  border: 5px solid gray;
  margin: 0;
}
```

CSS 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

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)

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`

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

Example

```
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;
}
```

CSS Text

CSS has a lot of properties for formatting text.

Text Color

The `color` property is used to set the color of the text. The color is specified by:

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

Look at CSS Color Values for a complete list of possible color values.

The default text color for a page is defined in the body selector.

Example

```
body {
  color: blue;
}

h1 {
  color: green;
}
```

Text Alignment and Text Direction

In this chapter you will learn about the following properties:

- `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):

Example

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

h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
```

Text Direction

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

Example

```
p {
  direction: rtl;
  unicode-bidi: bidi-override;
}
```

Vertical Alignment

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

Example

Set the vertical alignment of an image in a text:

```
img.a {
  vertical-align: baseline;
}
```

```css
img.b {
  vertical-align: text-top;
}

img.c {
  vertical-align: text-bottom;
}

img.d {
  vertical-align: sub;
}

img.e {
  vertical-align: super;
}
```

Text Decoration

In this chapter you will learn about the following properties:

- `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.

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

Example

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

```css
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.

Example

```css
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;
}
```

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:

Example

```
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:

Example

```
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:

Example

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

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

Line Height

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

Example

```
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:

Example

```
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:

Example

```
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!

Example

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

CSS Fonts

Choosing the right font for your website is important!

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

Example

Specify some different fonts for three paragraphs:

```
.p1 {
  font-family: "Times New Roman", Times, serif;
}

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

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
```

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.

Example

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)

Commonly Used Font Fallbacks

Below are some commonly used font fallbacks, organized by the 5 generic font families:

- **Serif**
- **Sans-serif**
- **Monospace**
- **Cursive**
- **Fantasy**

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)

Example

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

p.italic {
  font-style: italic;
}

p.oblique {
  font-style: oblique;
}
```

Font Weight

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

Example

```
p.normal {
  font-weight: normal;
}

p.thick {
  font-weight: bold;
}
```

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.

Example

```
p.normal {
  font-variant: normal;
}

p.small {
  font-variant: small-caps;
}
```

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

Set Font Size With Pixels

Setting the text size with pixels gives you full control over the text size:

Example

```
h1 {
  font-size: 40px;
}

h2 {
  font-size: 30px;
}

p {
  font-size: 14px;
}
```

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.

Example

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

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

The CSS Font Property

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

The `font` property is a shorthand property for:

- `font-style`
- `font-variant`
- `font-weight`
- `font-size/line-height`
- `font-family`

Note: The `font-size` and `font-family` values are required. If one of the other values is missing, their default value are used.

Example

Use `font` to set several font properties in one declaration:

```
p.a {
  font: 20px Arial, sans-serif;
}

p.b {
  font: italic small-caps bold 12px/30px Georgia, serif;
}
```

Font Awesome Icons

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

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

Styling Links

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

Example

```
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

Example

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

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

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

/* selected link */
a:active {
  color: blue;
}
```

CSS Lists

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

Different List Item Markers

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

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

Example

```
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;
}
```

CSS Tables

Collapse Table Borders

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

Example

```
table {
  border-collapse: collapse;
}
```

CSS Layout – The display Property

The `display` property is the most important CSS property for controlling layout.

The display Property

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

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`.

The `display` property is used to change the default display behavior of HTML elements.

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>

CSS Layout - The position Property

The `position` property specifies the type of positioning method used for an element (static, relative, fixed, absolute or sticky).

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:

This element has position: static;

Here is the CSS that is used:

Example

```
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.

This element has position: relative;

Here is the CSS that is used:

Example

```
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:

Example

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

This element has `position: fixed;`

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.

Here is a simple example:

This element has position: relative;

This element has position: absolute;

Here is the CSS that is used:

Example

```
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.

In this example, the sticky element sticks to the top of the page (`top: 0`), when you reach its scroll position.

Example

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

The z-index Property

The `z-index` property specifies the stack order of an element.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:

Example

```
img {
  position: absolute;
  left: 0px;
  top: 0px;
  z-index: -1;
}
```

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

CSS Layout - 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

In its simplest use, the `float` property can be used to wrap text around images.

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

When clearing floats, you should match the clear to the float: If an element is floated to the left, then you should clear to the left. Your floated element will continue to float, but the cleared element will appear below it on the web page.

CSS Layout - display: inline-block

The display: inline-block Value

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.

CSS Combinators

CSS Combinators

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:

Example

```
div p {
  background-color: yellow;
}
```

CSS Pseudo-classes

What are 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

Anchor Pseudo-classes

Links can be displayed in different ways:

Example

```
/* 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;
}
```

CSS - The :first-child Pseudo-class

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

Match the first <p> element

In the following example, the selector matches any <p> element that is the first child of any element:

Example

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

CSS Pseudo-elements

What are 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

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:

Example

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

The ::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:

Example

```
p::first-letter {
  color: #ff0000;
  font-size: xx-large;
}
```

CSS - The ::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:

Example

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

CSS - The ::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:

Example

```
h1::after {
  content: url(smiley.gif);
}
```

CSS - The ::marker Pseudo-element

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

The following example styles the markers of list items:

Example

```
::marker {
  color: red;
  font-size: 23px;
```

CSS - The ::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:

Example

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

CSS Opacity / Transparency

The `opacity` property specifies the opacity/transparency of an element.

Transparent Image

The `opacity` property can take a value from 0.0 - 1.0. The lower the value, the more transparent.

Example

```
img {
  opacity: 0.5;
}
```

CSS Vertical Navigation Bar

To build a vertical navigation bar, you can style the <a> elements inside the list, in addition to the code from the previous page:

Example

```
li a {
  display: block;
  width: 60px;
}
```

Example explained:

- `display: block;` - Displaying the links as block elements makes the whole link area clickable (not just the text), and it allows us to specify the width (and padding, margin, height, etc. if you want)
- `width: 60px;` - Block elements take up the full width available by default. We want to specify a 60 pixels width

You can also set the width of <ul>, and remove the width of <a>, as they will take up the full width available when displayed as block elements. This will produce the same result as our previous example:

Horizontal Navigation Bar

There are two ways to create a horizontal navigation bar.
Using **inline** or **floating** list items.

Inline List Items

One way to build a horizontal navigation bar is to specify the <li> elements as inline, in addition to the "standard" code from the previous page:

Example

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

Example explained:

- `display: inline;` - By default, <li> elements are block elements. Here, we remove the line breaks before and after each list item, to display them on one line

Basic Dropdown

Create a dropdown box that appears when the user moves the mouse over an element.

Example

```
<style>
.dropdown {
  position: relative;
  display: inline-block;
}
```

```
.dropdown-content {
  display: none;
  position: absolute;
  background-color: #f9f9f9;
  min-width: 160px;
  box-shadow: 0px 8px 16px 0px rgba(0,0,0,0.2);
  padding: 12px 16px;
  z-index: 1;
}

.dropdown:hover .dropdown-content {
  display: block;
}
</style>

<div class="dropdown">
  <span>Mouse over me</span>
  <div class="dropdown-content">
    <p>Hello World!</p>
  </div>
</div>
```

Example Explained

HTML) Use any element to open the dropdown content, e.g. a , or a <button> element.

Use a container element (like) to create the dropdown content and add whatever you want inside of it.

Wrap a element around the elements to position the dropdown content correctly with CSS.

CSS) The `.dropdown` class uses `position:relative`, which is needed when we want the dropdown content to be placed right below the dropdown button (using `position:absolute`).

The `.dropdown-content` class holds the actual dropdown content. It is hidden by default, and will be displayed on hover (see below). Note the `min-width` is set to 160px. Feel free to change this. **Tip:** If you want the width of the dropdown content to be as wide as the dropdown button, set the `width` to 100% (and `overflow:auto` to enable scroll on small screens).

Instead of using a border, we have used the CSS `box-shadow` property to make the dropdown menu look like a "card".

The `:hover` selector is used to show the dropdown menu when the user moves the mouse over the dropdown button.

CSS Attribute Selectors

Style HTML Elements With Specific Attributes

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

CSS [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:

Example

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

CSS [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:

Example

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

CSS [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":

Example

```
[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".

CSS [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".

Example

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

CSS [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!

Example

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

CSS [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!

Example

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

CSS [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!

Example

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

CSS Specificity

What is 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:

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

CSS The !important Rule

What is !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!

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!

CSS 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