Co-funded by the
Erasmus+ Programme
of the European Union

3.1.:

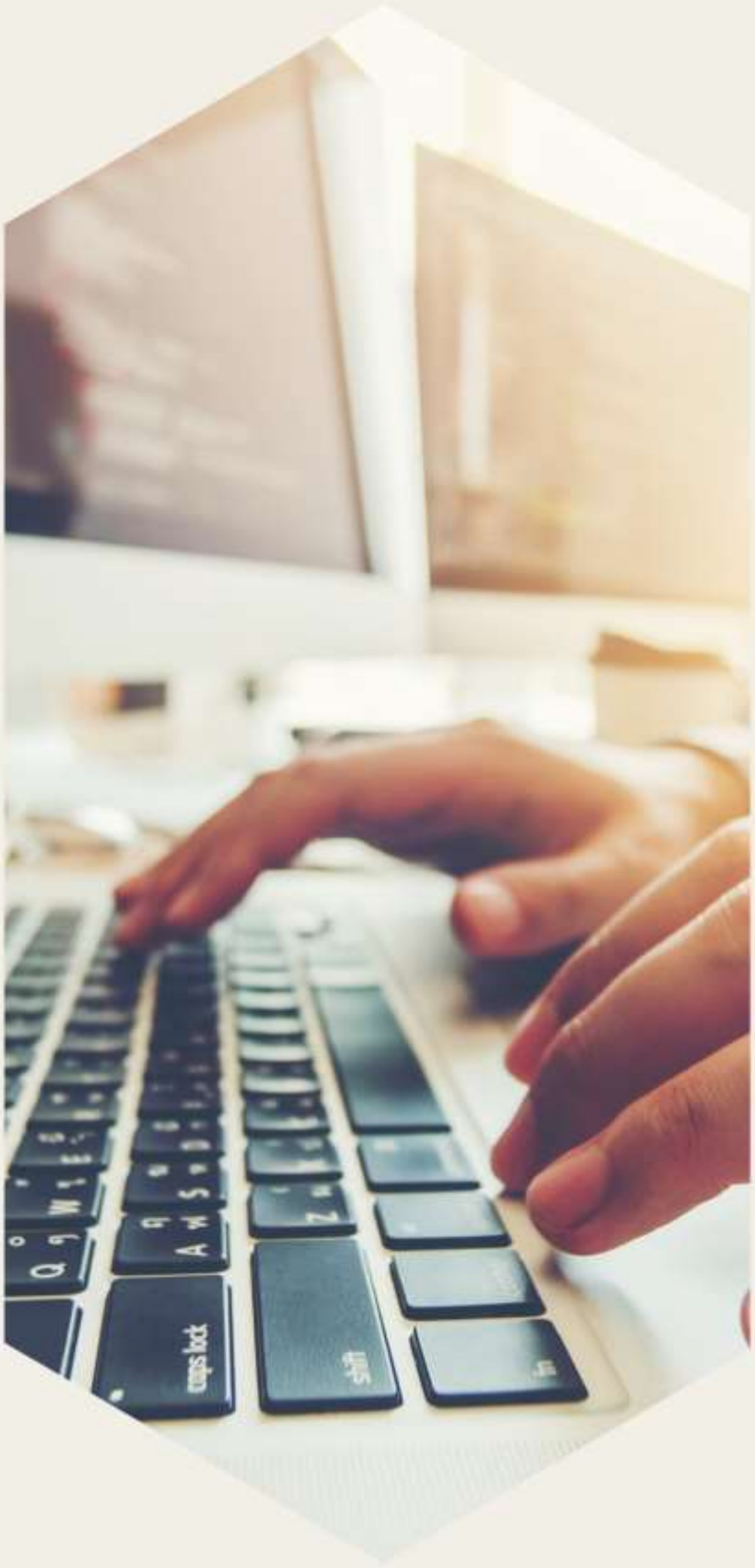
Code4SP Training Material Package

WP3:

Code4SP Training Materials – CSS

Prepared by:

social hackers academy



Project Information

Project Acronym: Code4SP

Project Title: Coding for Social Promotion

Project Reference: 621417-EPP-1-2020-1-PT-EPPKA3-IPI-SOC-IN

Project website: www.code4sp.eu

Authoring Partner: SHA

Document Version: 2

Date of Preparation: 11/03/2022

Document History			
Date	Version	Author	Description
11/03/2022	1	SHA	Draft
22/03/2022	2	CEPROF	Revision
28/03/2022	3	SHA	Revision

Table of Content

Topic Information

Co-funded by the Erasmus+ Programme of the European Union

Topic:

3. CSS

Prerequisites:

Basic computer literacy, basic software installed, basic knowledge of working with files, and HTML basics (study Introduction to HTML.)

Workload:

10 hours.

Description:

Cascading Style Sheets (CSS) is a stylesheet language used to describe the presentation of a document written in HTML or XML (including XML dialects such as SVG, MathML or XHTML). CSS describes how elements should be rendered on screen, on paper, in speech, or on other media.

Learning outcomes:

Learners will get knowledge on how to define CSS, use basic CSS syntax, set up web pages with CSS, use CSS for styling text, font, and properties, use CSS for styling page backgrounds, style lists in CSS, use CSS classes and IDs, use borders and height and width CSS properties, use CSS pseudo elements, position elements with CSS and validate CSS and HTML.

Material required:

- Computer or laptop
- Internet connection
- Text editor (online or offline): Sublime Text/Brackets/W3Schools online editor

Lesson Scenario:

The total time for this topic is 10 hours, and it will be up to the trainer/coach to decide how much time to dedicate to teaching each subtopic. To make the most of all the time available, we propose the use of the training materials produced by the project (PPT presentations), which were designed with an effective use of time in mind. These presentations are composed of the following elements:

- Development of the subtopic and main ideas to retain;
- Proposed Activities/Exercises.

That said, if the trainer/coach follows the logical sequence of the PPTs, he/she will certainly be able to complete the session within the stipulated time limit. These presentations can also be made available to learners for individual study.

Subtopics:

- CSS Intro
- CSS Responsive Web Design
- CSS GRID
- CSS Advanced
- SASS

Additional resources:

- CSS tutorial: w3schools
- Online Course on HTML and CSS: CodeAcademy

Co-funded by the Erasmus+ Programme of the European Union

CSS Intro

What is CSS?

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, as well as a variety of other effects.

CSS is easy to learn and understand but it provides a powerful control over the presentation of an HTML document. Most commonly, CSS is combined with the markup languages HTML or XHTML.

Advantages of CSS

- **CSS saves time** - You can write CSS once and then reuse the same sheet in multiple HTML pages. You can define a style for each HTML element and apply it to as many web pages as you want.
- **Pages load faster** - If you are using CSS, you do not need to write HTML tag attributes every time. Just write one CSS rule of a tag and apply it to all the occurrences of that tag. So, less code means faster download times.
- **Easy maintenance** - To make a global change, simply change the style, and all the elements in all the web pages will be updated automatically.

- **Superior styles to HTML** - CSS has a much wider array of attributes than HTML, so you can give a far better look to your HTML page in comparison to HTML attributes.
- **Multiple Device Compatibility** - Style sheets allow content to be optimized for more than one type of device. By using the same HTML document, different versions of a website can be presented for handheld devices such as PDAs and cellphones or for printing.
- **Global web standards** – Now HTML attributes are being deprecated and it is being recommended to use CSS. So it's a good idea to start using CSS in all the HTML pages to make them compatible with future browsers.

Who Creates and Maintains CSS?

CSS is created and maintained through a group of people within the W3C called the CSS Working Group. The CSS Working Group creates documents called specifications. When a specification has been discussed and officially ratified by the W3C members, it becomes a recommendation.

These ratified specifications are called recommendations because the W3C has no control over the actual implementation of the language. Independent companies and organizations create that software.

NOTE: The World Wide Web Consortium or W3C is a group that makes recommendations about how the Internet works and how it should evolve.

CSS Versions

CSS was originally released in 1996 and consists of properties for adding font properties such as typeface and emphasis color of text, backgrounds, and other elements. CSS2 was released in 1998 with added styles for other media types so that it can be used for page layout designing. CSS3 was released in 1999 and presentation-style properties were added in it that allows you to build a presentation from documents.

CSS Syntax

A CSS comprises of style rules that are interpreted by the browser and then applied to the corresponding elements in your document. A style rule is made of three parts:

- **Selector**: A selector is an HTML tag at which a style will be applied. This could be any tag like <h1> or <table> etc.
- **Property**: A property is a type of attribute of HTML tag. Put simply, all the HTML attributes are converted into CSS properties. They could be color, border, etc.
- **Value**: Values are assigned to properties. For example, color property can have the value either red or #F1F1F1 etc.

CSS Selectors

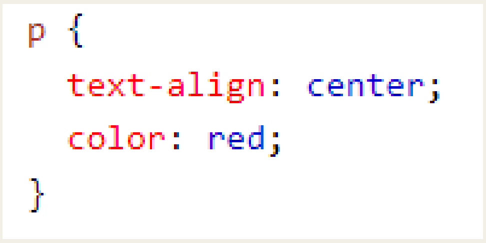
CSS selectors are used to "find" (or select) the HTML elements you want to style.

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;
}
```

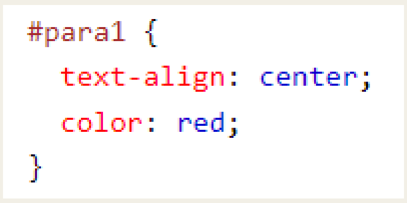
Figure 1 *– The CSS element selector (**Source**: https://www.w3schools.com/css/css_selectors.asp)*

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.



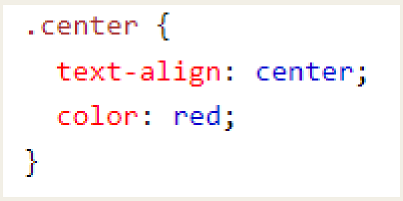
```
#para1 {
  text-align: center;
  color: red;
}
```

Figure 2 *– The CSS ID Selector (**Source**: https://www.w3schools.com/css/css_selectors.asp)*

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.

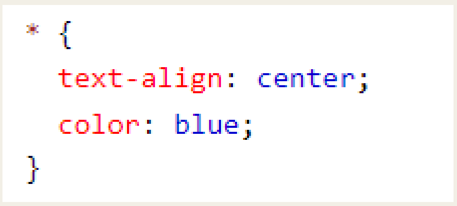


```
.center {
  text-align: center;
  color: red;
}
```

Figure 3 *– The CSS class Selector (**Source**: https://www.w3schools.com/css/css_selectors.asp)*

The CSS universal Selector

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

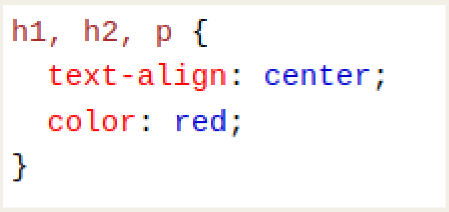


```
* {
  text-align: center;
  color: blue;
}
```

Figure 5 *– The CSS universal Selector (**Source**: https://www.w3schools.com/css/css_selectors.asp)*

Grouping selectors

You can apply a style to many selectors if you like. Just separate the selectors with a comma, as given in the following example:



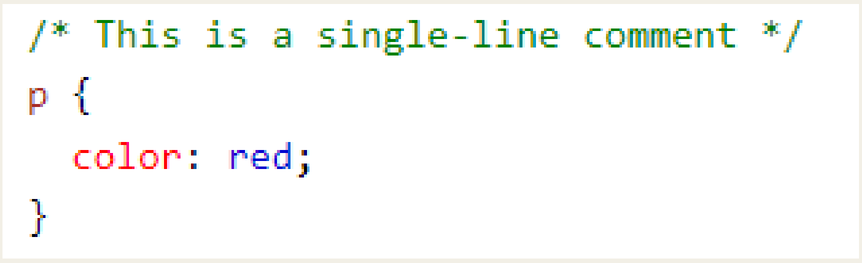
```
h1, h2, p {
  text-align: center;
  color: red;
}
```

Figure 6 *– Grouping Selectors (**Source**: https://www.w3schools.com/css/css_selectors.asp)*

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 starts with /* and ends with */



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

Figure 7 *– CSS comments (**Source**: https://www.w3schools.com/css/css_comments.asp)*

CSS Colors

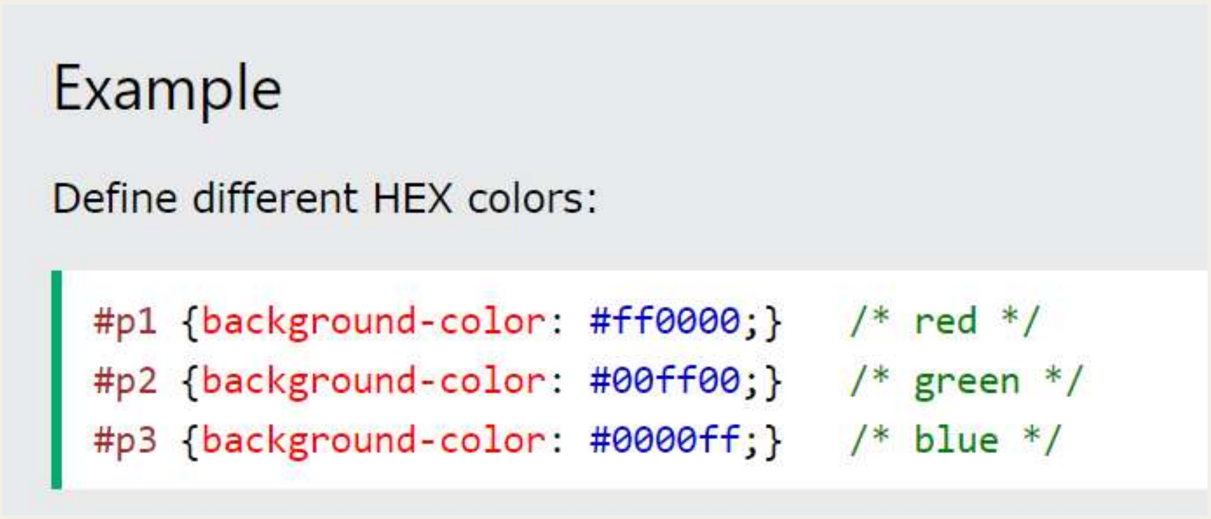
Colors in CSS can be specified by the following methods:

- Hexadecimal colors
- Hexadecimal colors with transparency
- RGB colors
- RGBA colors
- HSL colors
- HSLA colors
- Predefined/Cross-browser color names
- With the current color keyword

Hexadecimal 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. All values must be between 00 and FF.

For example, the #0000ff value is rendered as blue, because the blue component is set to its highest value (ff) and the others are set to 00.



Example

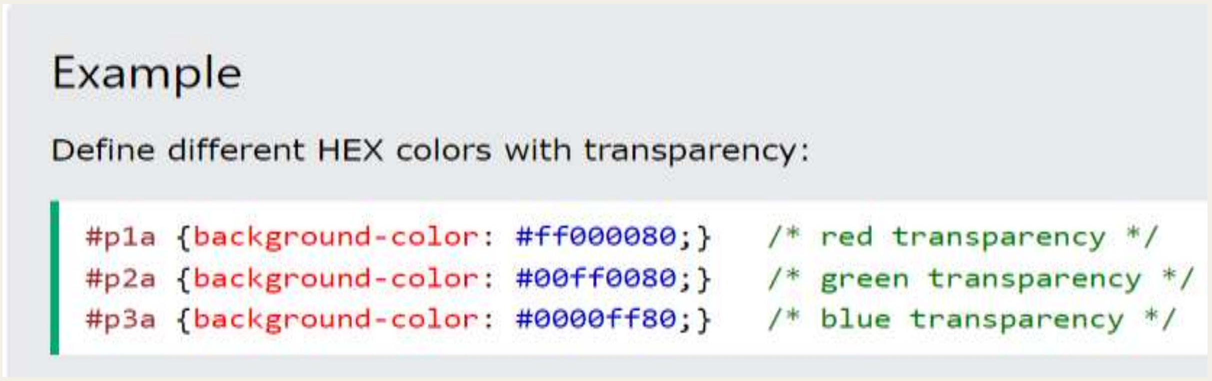
Define different HEX colors:

```
#p1 {background-color: #ff0000;}   /* red */
#p2 {background-color: #00ff00;}   /* green */
#p3 {background-color: #0000ff;}   /* blue */
```

Figure 8 *– Hex Colors (**Source**: https://www.w3schools.com/css/css_colors.asp)*

Hexadecimal Colors With Transparency

A hexadecimal color is specified with: #RRGGBB. To add transparency, add two additional digits between 00 and FF.



Example

Define different HEX colors with transparency:

```
#p1a {background-color: #ff000080;}   /* red transparency */
#p2a {background-color: #00ff0080;}   /* green transparency */
#p3a {background-color: #0000ff80;}   /* blue transparency */
```

***Figure 9** – Hex Colors with transparency (**Source**: https://www.w3schools.com/css/css_colors.asp)*

RGB Colors

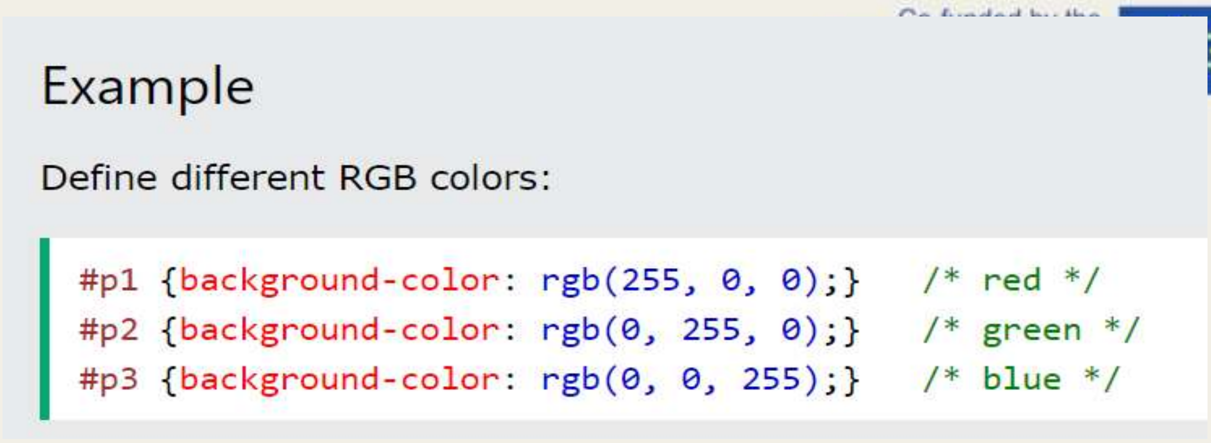
An RGB color value is specified with the rgb() function, which has the following syntax:

rgb(red, green, blue)

Each parameter (red, green, and blue) defines the intensity of the color and can be an integer between 0 and 255 or a percentage value (from 0% to 100%).

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

Also, the following values define equal color: rgb(0,0,255) and rgb(0%,0%,100%).


Example

Define different RGB colors:

```
#p1 {background-color: rgb(255, 0, 0);}    /* red */
#p2 {background-color: rgb(0, 255, 0);}    /* green */
#p3 {background-color: rgb(0, 0, 255);}    /* blue */
```

Figure 10 *– RGB Colors (**Source**: https://www.w3schools.com/css/css_colors.asp)*

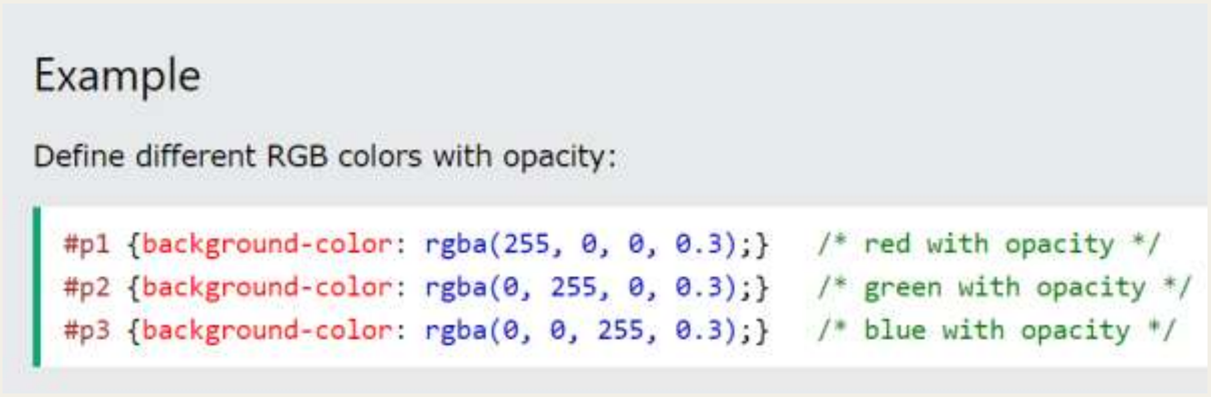
RGBA Colors

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity of the object.

An RGBA color is specified with the rgba() function, which has the following syntax:

rgba(red, green, blue, alpha)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).


Example

Define different RGB colors with opacity:

```
#p1 {background-color: rgba(255, 0, 0, 0.3);}   /* red with opacity */
#p2 {background-color: rgba(0, 255, 0, 0.3);}   /* green with opacity */
#p3 {background-color: rgba(0, 0, 255, 0.3);}   /* blue with opacity */
```

Figure 11 *– RGB Colors with opacity (**Source**: https://www.w3schools.com/css/css_colors.asp)*

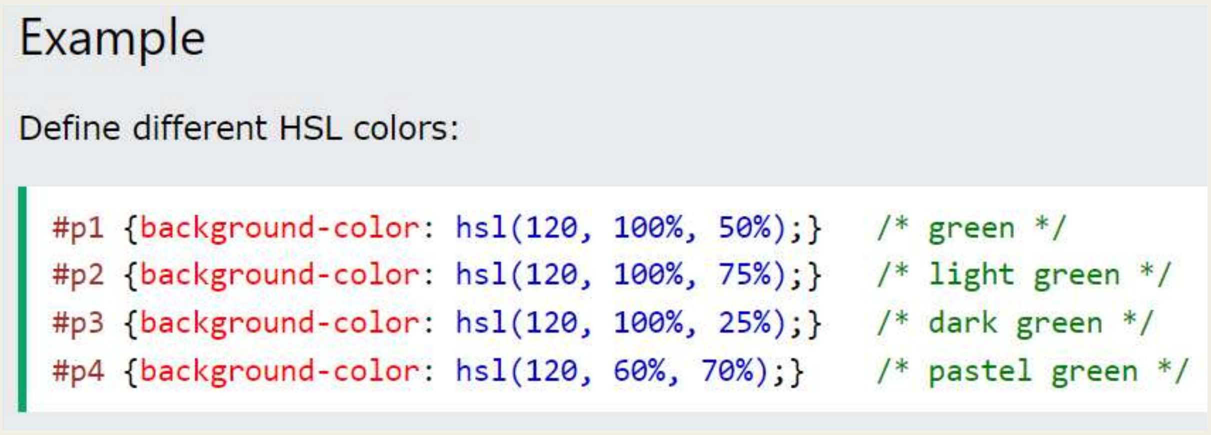
HSL Colors

HSL stands for hue, saturation, and lightness - and represents a cylindrical-coordinate representation of colors.

An HSL color value is specified with the hsl() function, which has the following syntax:

hsl(hue, saturation, lightness)

Hue is a degree on the color wheel (from 0 to 360) - 0 (or 360) is red, 120 is green, 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, 100% is white.


Example

Define different HSL colors:

```
#p1 {background-color: hsl(120, 100%, 50%);}   /* green */
#p2 {background-color: hsl(120, 100%, 75%);}   /* light green */
#p3 {background-color: hsl(120, 100%, 25%);}   /* dark green */
#p4 {background-color: hsl(120, 60%, 70%);}    /* pastel green */
```

***Figure 12** – HSL Colors (**Source**: https://www.w3schools.com/css/css_colors.asp)*

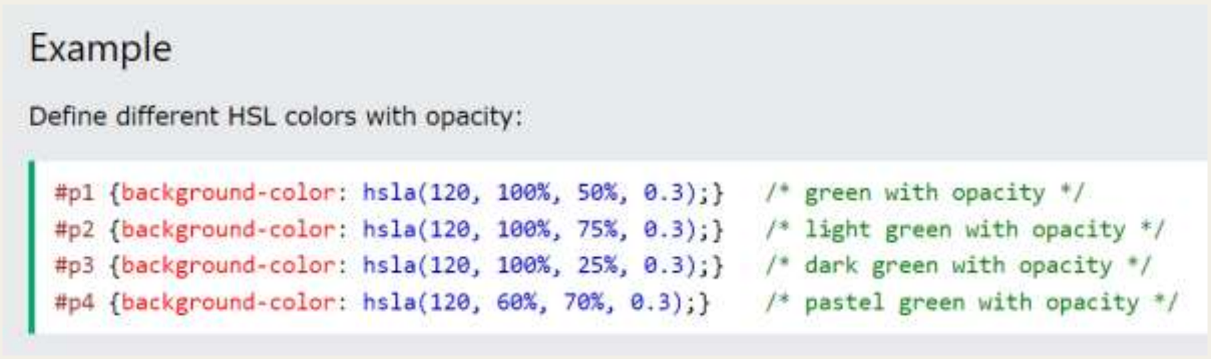
HSLA Colors

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

An HSLA color value is specified with the hsla() function, which has the following syntax:

hsla(hue, saturation, lightness, alpha)

The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).


Example

Define different HSL colors with opacity:

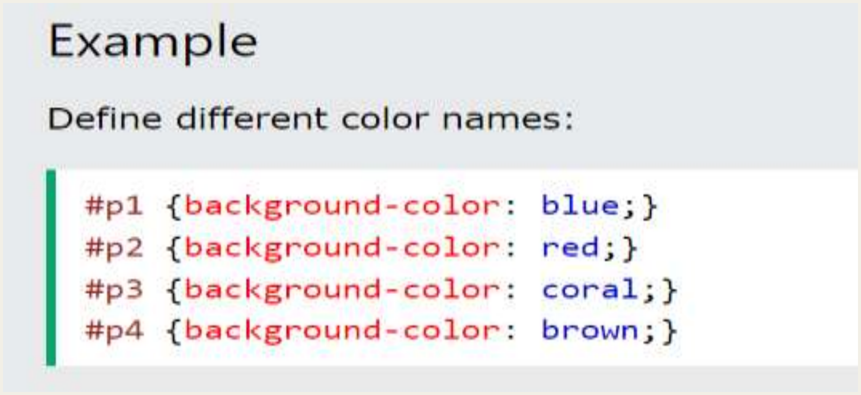
```
#p1 {background-color: hsla(120, 100%, 50%, 0.3);}   /* green with opacity */
#p2 {background-color: hsla(120, 100%, 75%, 0.3);}   /* light green with opacity */
#p3 {background-color: hsla(120, 100%, 25%, 0.3);}   /* dark green with opacity */
#p4 {background-color: hsla(120, 60%, 70%, 0.3);}    /* pastel green with opacity */
```

Figure 13 *– HSL Colors with opacity (**Source**: https://www.w3schools.com/css/css_colors.asp)*

Predefined/Cross-browser Color Names

140 color names are predefined in the HTML and CSS color specification.

For example: blue, red, coral, brown, etc:


Example

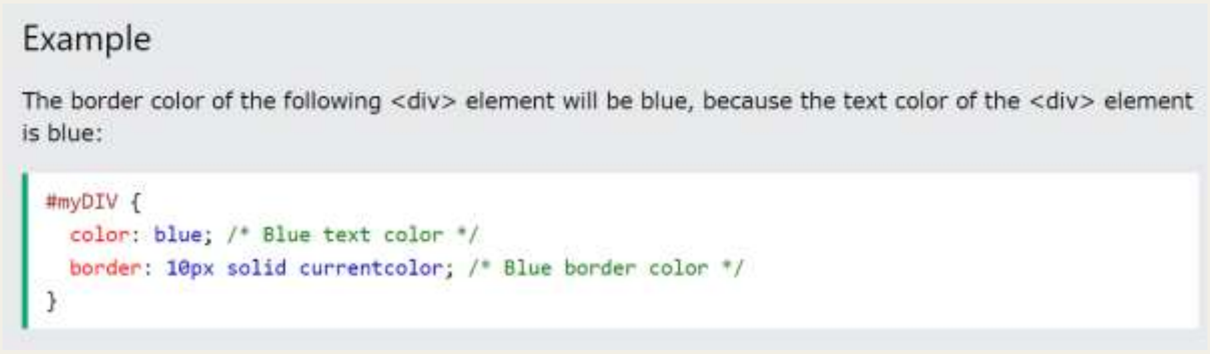
Define different color names:

```
#p1 {background-color: blue;}
#p2 {background-color: red;}
#p3 {background-color: coral;}
#p4 {background-color: brown;}
```

Figure 14 *– Cross-browser Colors (**Source**: https://www.w3schools.com/css/css_colors.asp)*

The currentcolor Keyword

The currentcolor keyword refers to the value of the color property of an element.



Example

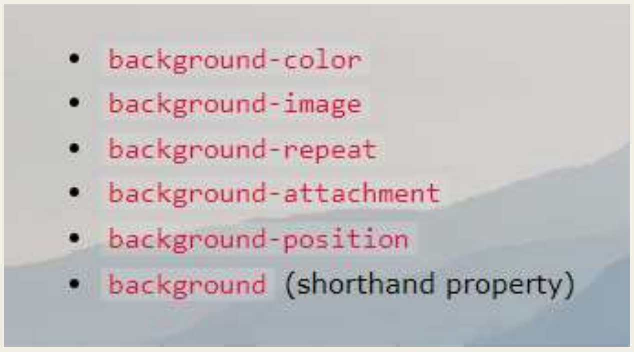
The border color of the following <div> element will be blue, because the text color of the <div> element is blue:

```
#myDIV {
  color: blue; /* Blue text color */
  border: 10px solid currentcolor; /* Blue border color */
}
```

***Figure 15** – The currentcolor keyword (**Source**: https://www.w3schools.com/colors/colors_currentcolor.asp)*

CSS Backgrounds

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

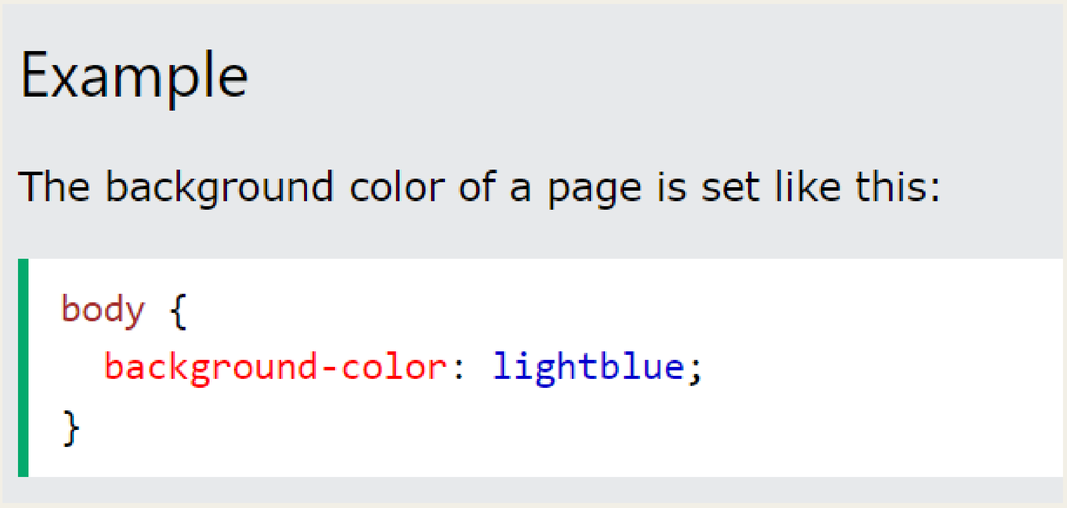


- background-color
- background-image
- background-repeat
- background-attachment
- background-position
- background (shorthand property)

***Figure 16** – Background (**Source**: https://www.w3schools.com/css/css_background.asp)*

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:

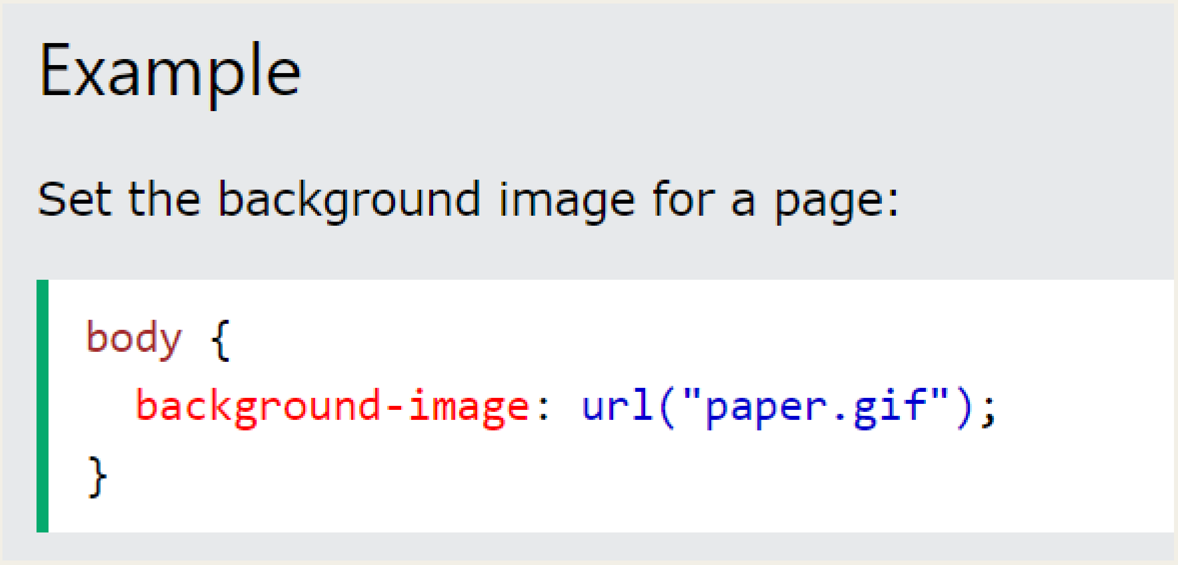
```
body {
  background-color: lightblue;
}
```

***Figure 17** – Background color (**Source**:https://www.w3schools.com/css/css_background.asp)*

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");
}
```

***Figure 18** – Background image (**Source**: https://www.w3schools.com/css/css_background.asp)*

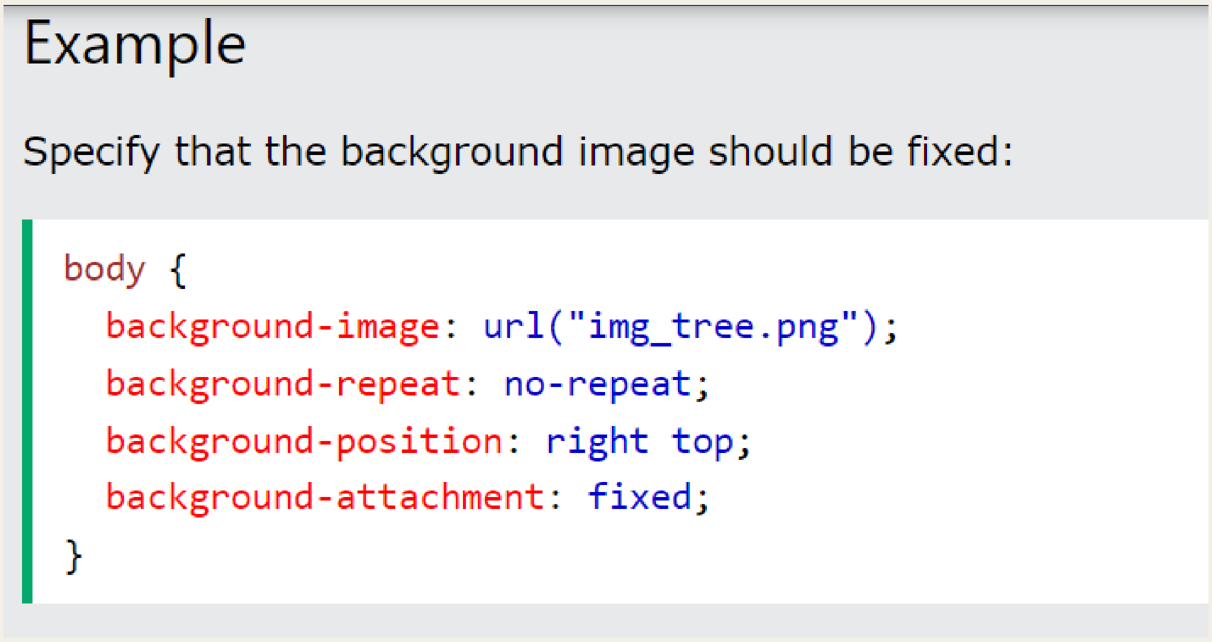
CSS background-repeat

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.

CSS background-attachment

The background-attachment property specifies whether the background image should scroll or be fixed

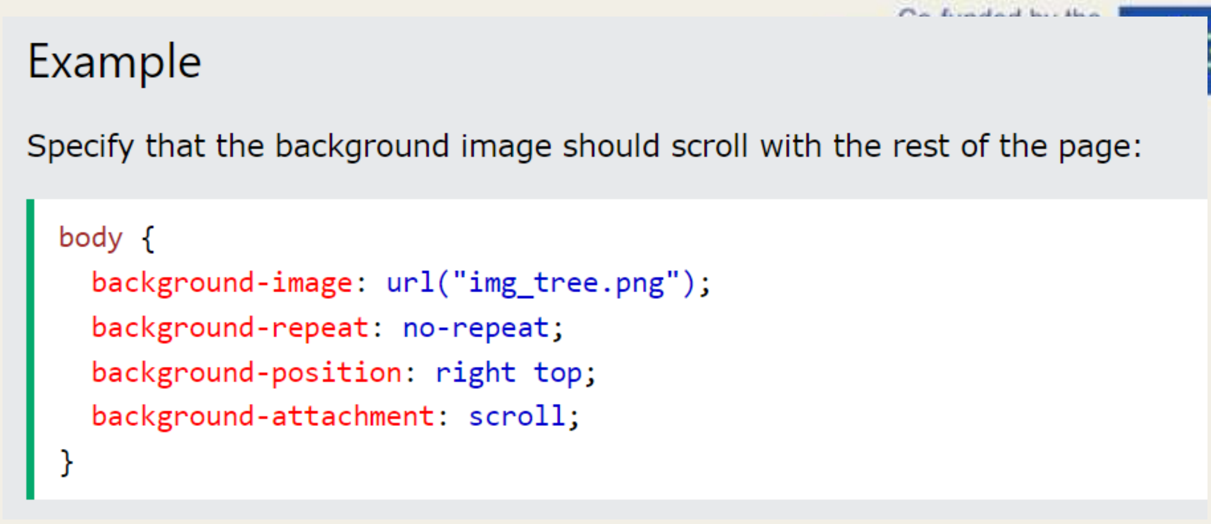


Example

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;
}
```

***Figure 19** – Background-attachement (**Source**: https://www.w3schools.com/css/css_background_attachment.asp)*


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;
}
```

Figure 20 *– Background-attachement (**Source**: https://www.w3schools.com/css/css_background_attachment.asp)*

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:

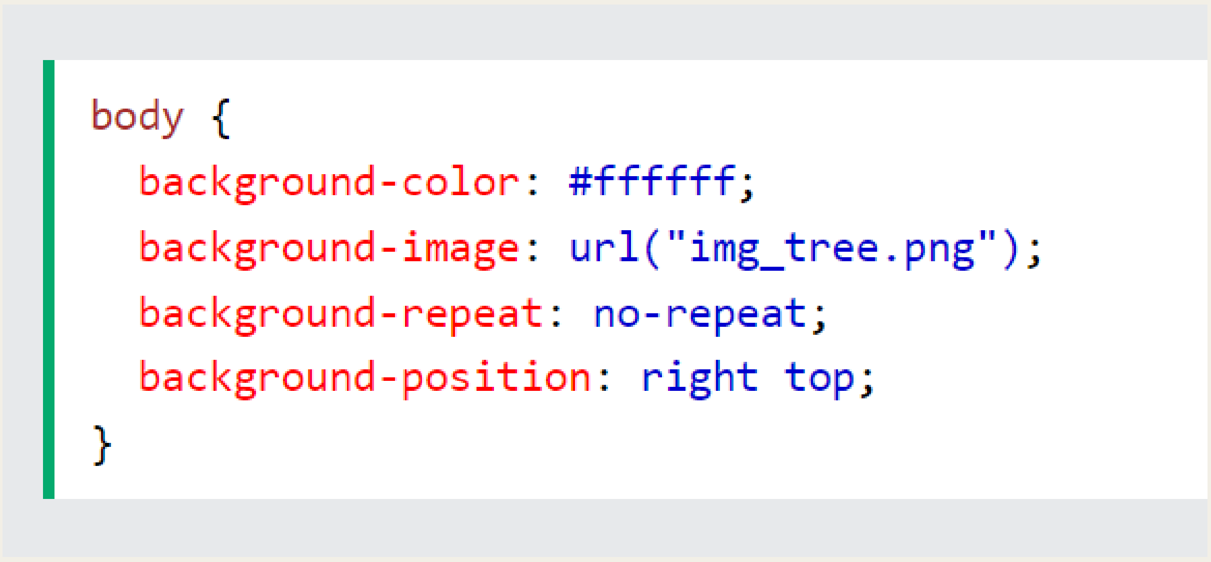
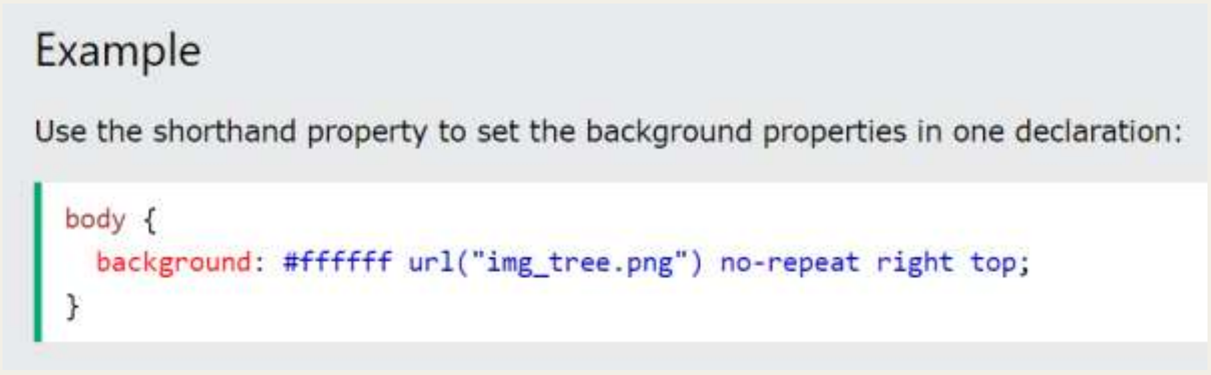

```
body {
  background-color: #ffffff;
  background-image: url("img_tree.png");
  background-repeat: no-repeat;
  background-position: right top;
}
```

Figure 21 *– Background shorthand property (**Source**:https://www.w3schools.com/css/css_background_shorthand.asp)*

You can use the shorthand property background:



Example

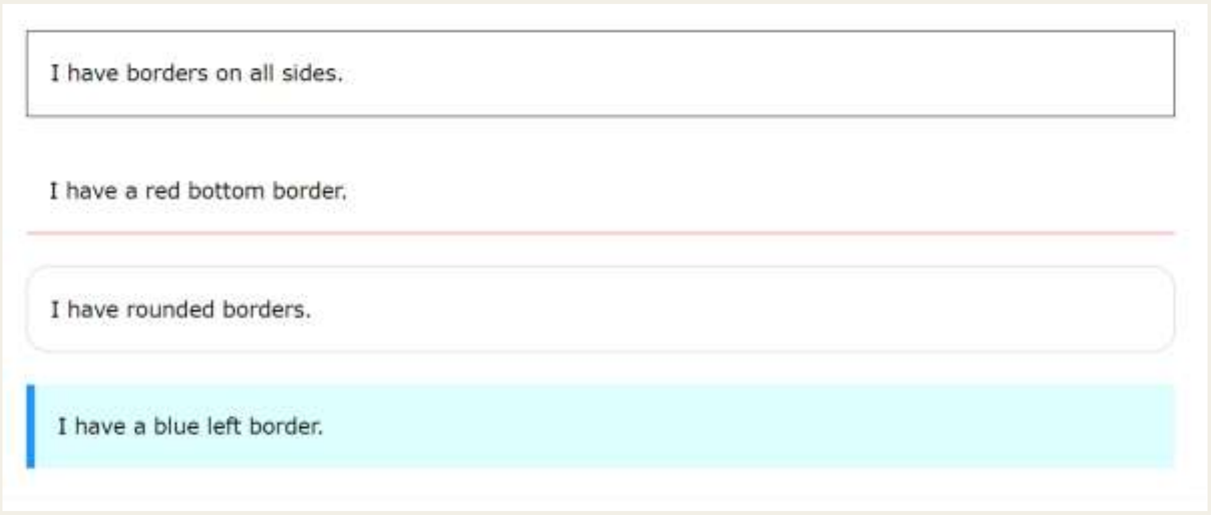
Use the shorthand property to set the background properties in one declaration:

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

***Figure 22** – Background shorthand property*
*(**Source**:https://www.w3schools.com/css/css_background_shorthand.asp)*

CSS Border

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



I have borders on all sides.

I have a red bottom border.

I have rounded borders.

I have a blue left border.

***Figure 23** – Border (**Source**:https://www.w3schools.com/css/css_border.asp)*

CSS Margins

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

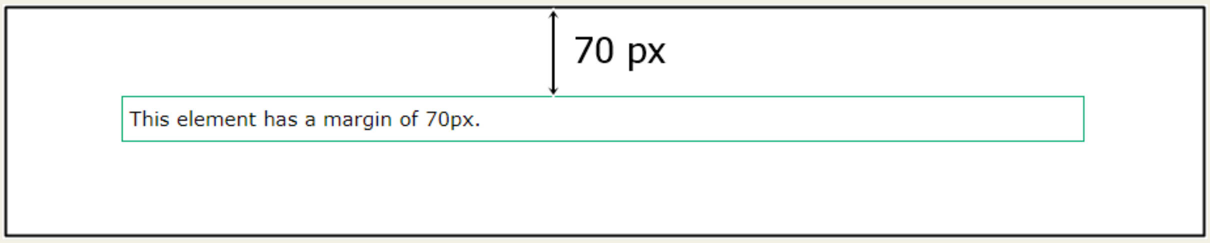
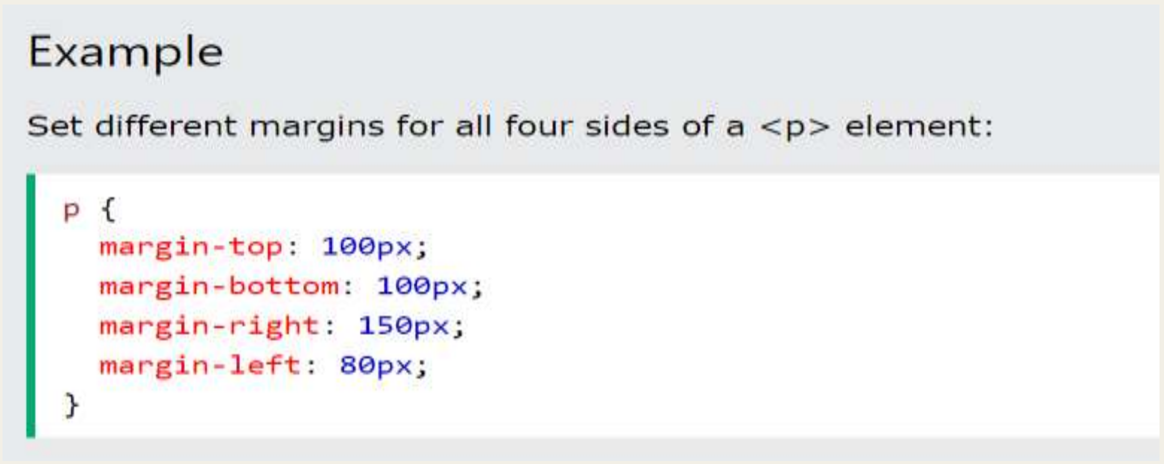


Figure 24 *– Margin (**Source**:https://www.w3schools.com/css/css_margin.asp)*

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).



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;
}
```

Figure 25 *– Margin (**Source**:https://www.w3schools.com/css/css_margin.asp)*

CSS Padding

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

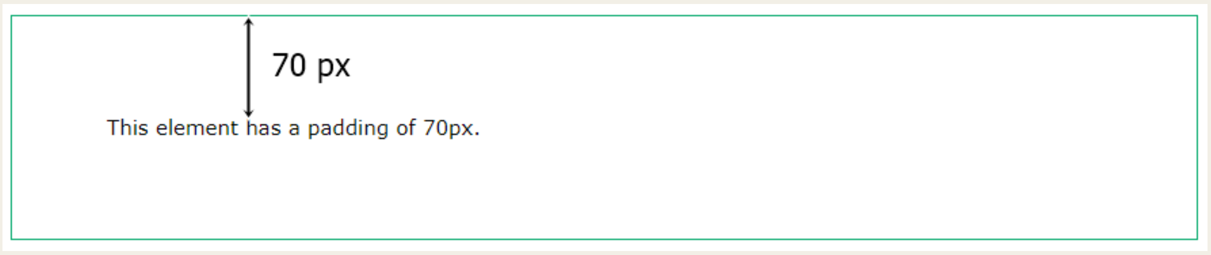
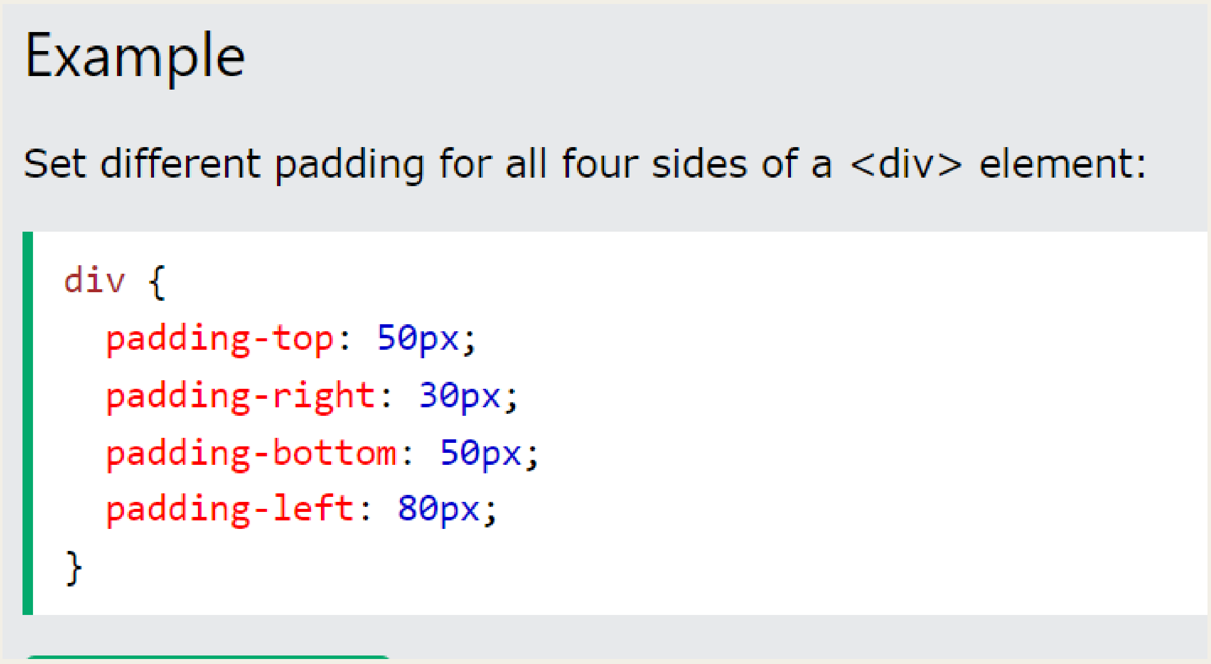


Figure 26 *– Padding (**Source**:https://www.w3schools.com/css/css_padding.asp)*

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).



Example

Set different padding for all four sides of a <div> element:

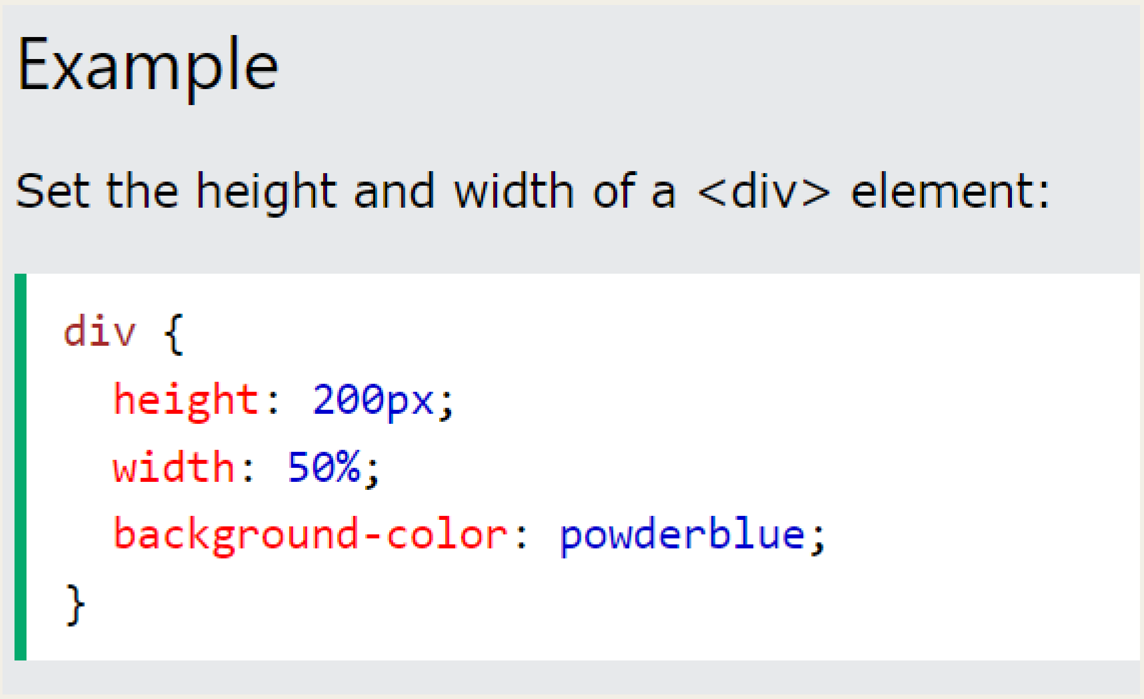
```
div {
  padding-top: 50px;
  padding-right: 30px;
  padding-bottom: 50px;
  padding-left: 80px;
}
```

Figure 27 *– Padding (**Source**:https://www.w3schools.com/css/css_padding.asp)*

CSS 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. 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 <div> element:

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

Figure 28 – *Height and width (**Source**: https://www.w3schools.com/css/css_dimension.asp)*

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:

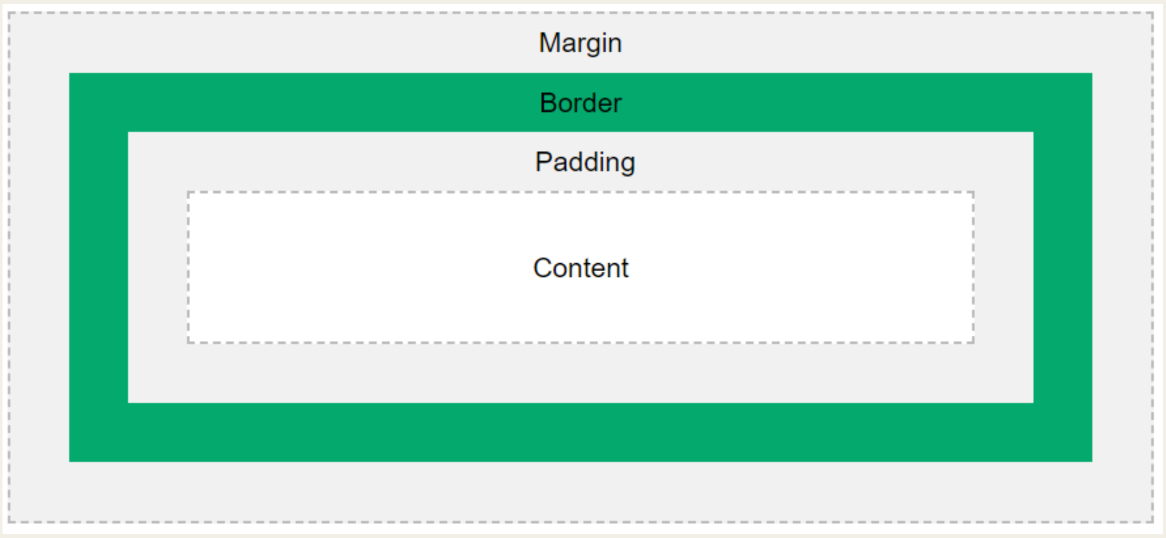


Figure 29 *– Box model (**Source**: https://www.w3schools.com/css/css_boxmodel.asp)*

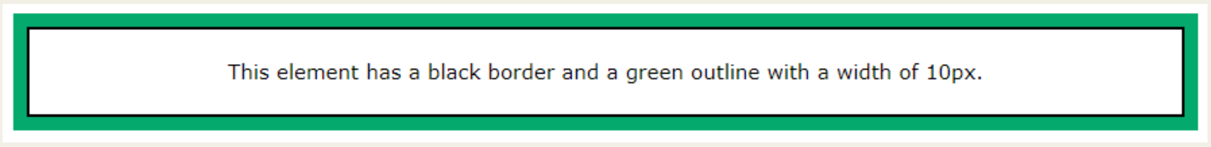
Explanation of the different parts:

- 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.

CSS Outline

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

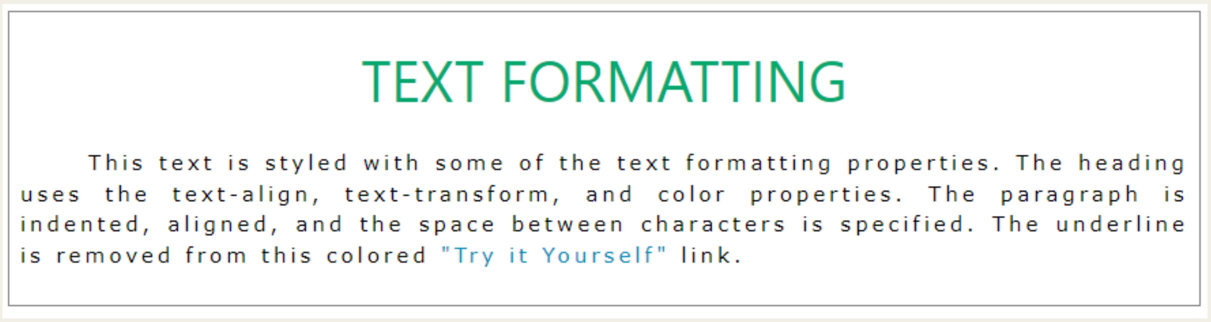


This element has a black border and a green outline with a width of 10px.

Figure 30 *– CSS Outline (**Source**: https://www.w3schools.com/css/css_outline.asp)*

CSS Text

CSS has a lot of properties for formatting text.



TEXT FORMATTING

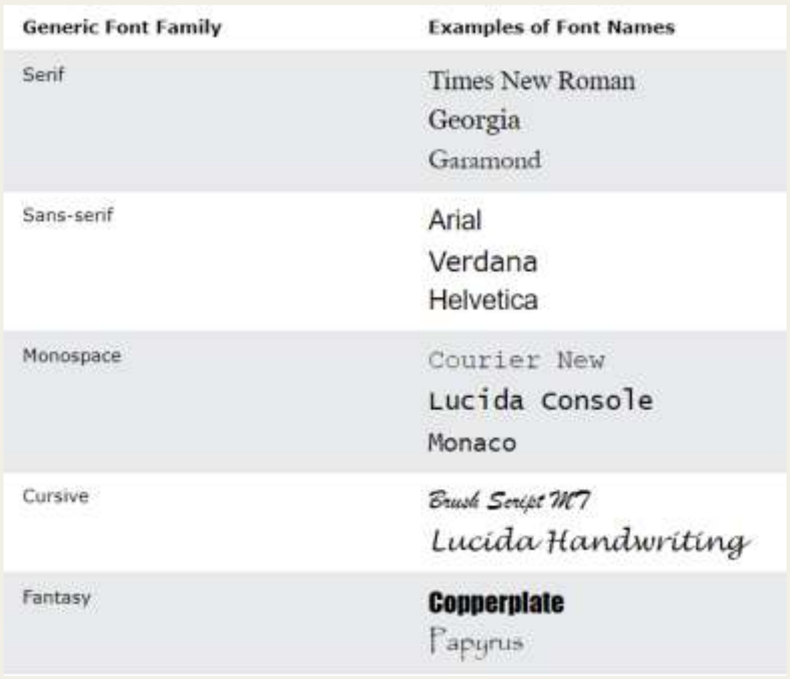
This text is styled with some of the text formatting properties. The heading uses the text-align, text-transform, and color properties. The paragraph is indented, aligned, and the space between characters is specified. The underline is removed from this colored "Try it Yourself" link.

Figure 31 *– CSS Text (**Source**: https://www.w3schools.com/css/css_text.asp)*

- Text Color
- Text Alignment
- Text Decoration
- Text Transformation
- Text Spacing
- Text Shadow

CSS Fonts

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

Figure 32 *– CSS Fonts (**Source**: https://www.w3schools.com/css/css_font.asp)*

CSS Icons

- The simplest way to add an icon to your HTML page, is with an icon library, such as Font Awesome.

- Add the name of the specified icon class to any inline HTML element (like <i> or <span>).
- All the icons in the icon libraries below, are scalable vectors that can be customized with CSS

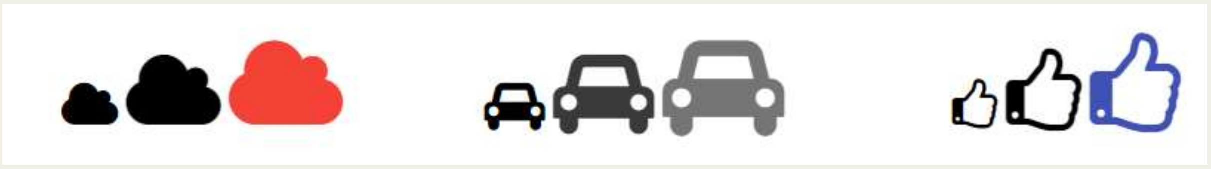


***Figure 33** – CSS Icons (**Source**: https://www.w3schools.com/css/css_icons.asp)*

CSS Links

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

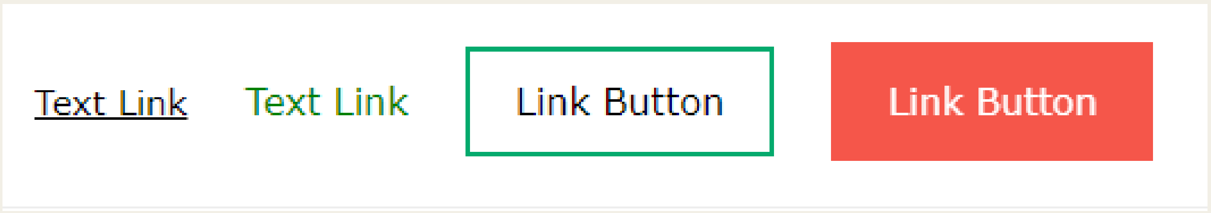


***Figure 34** – Links (**Source**: https://www.w3schools.com/css/css_link.asp)*

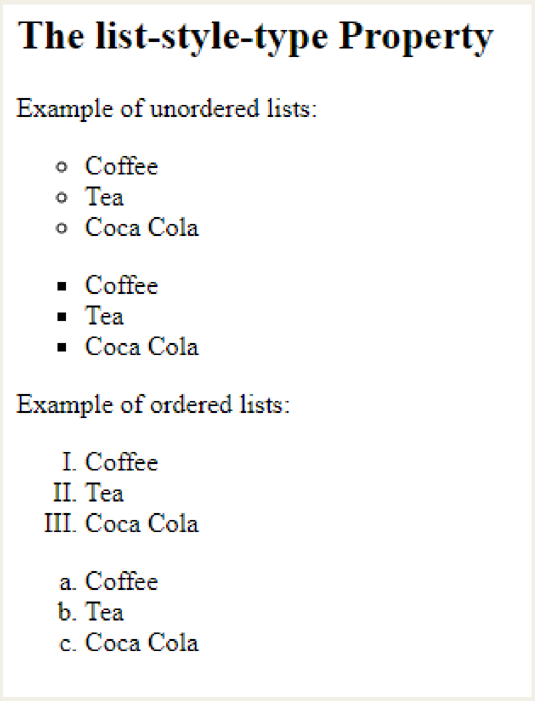
In addition, links can be styled differently depending on what state they are in.
The four links states are:

- 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

CSS Lists

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
-



The list-style-type Property

Example of unordered lists:

- Coffee
- Tea
- Coca Cola

- Coffee
- Tea
- Coca Cola

Example of ordered lists:

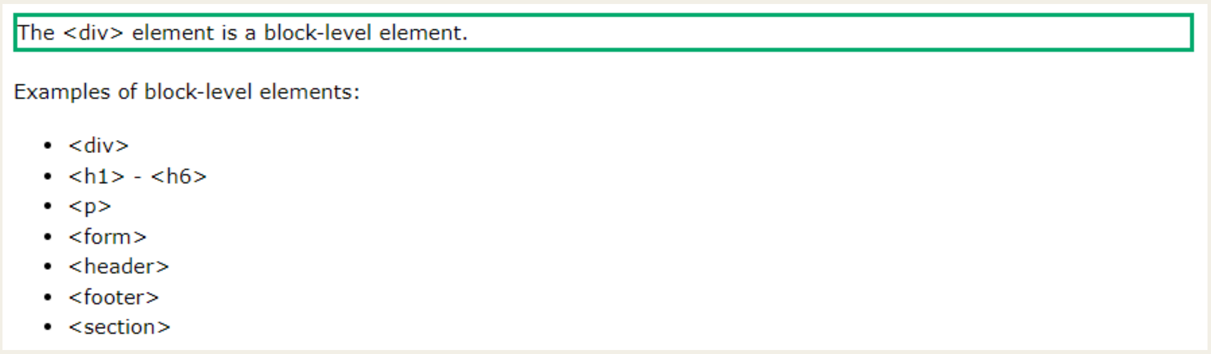
I. Coffee
II. Tea
III. Coca Cola

a. Coffee
b. Tea
c. Coca Cola

***Figure 35** – Lists (**Source**: https://www.w3schools.com/css/css_list.asp)*

CSS Display

- The display property specifies if/how an element is displayed.
- Every HTML element has a default display value depending on what type of element it is. The default display value for most elements is block or inline



The <div> element is a block-level element.

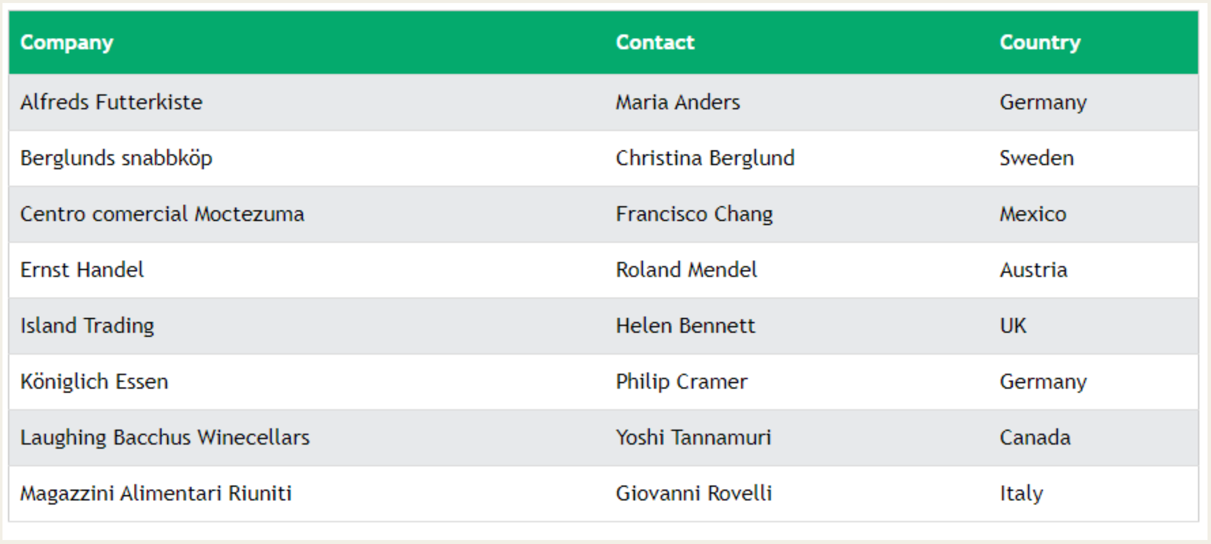
Examples of block-level elements:

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

Figure 36 *– Display (**Source**: https://www.w3schools.com/css/css_display_visibility.asp)*

CSS Tables

The look of an HTML table can be greatly improved with CSS:



Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Ernst Handel	Roland Mendel	Austria
Island Trading	Helen Bennett	UK
Königlich Essen	Philip Cramer	Germany
Laughing Bacchus Winecellars	Yoshi Tannamuri	Canada
Magazzini Alimentari Riuniti	Giovanni Rovelli	Italy

Figure 37 *– Tables (**Source**: https://www.w3schools.com/css/css_table.asp)*

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

CSS Max-width

- The problem with the <div> above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page
- Using max-width instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices.

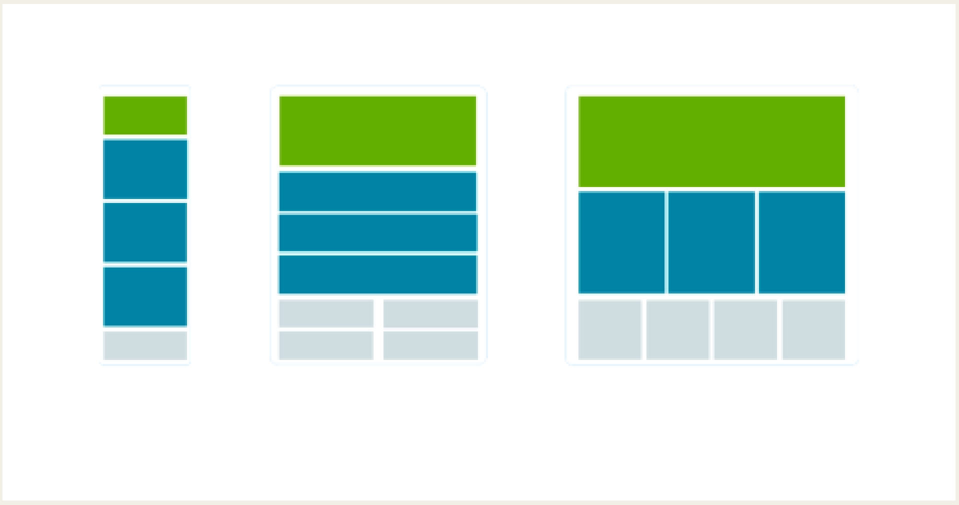


Figure 38 *– Max Width (**Source**: https://www.w3schools.com/css/css_max-width.asp)*

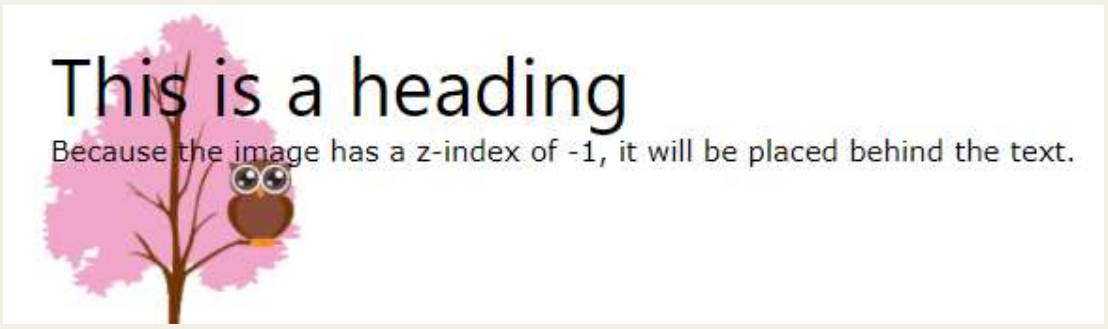
CSS Position

- The position property specifies the type of positioning method used for an element
- There are five different position values:
- static
- relative
- fixed
- absolute
- sticky

CSS Z-index

- 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:


This is a heading

Because the image has a z-index of -1, it will be placed behind the text.

Figure 39 *– Z-index (**Source**: https://www.w3schools.com/css/css_z-index.asp)*

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

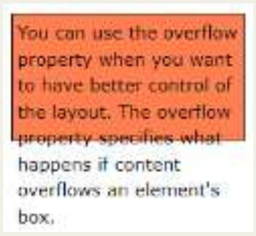

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what happens if content overflows an element's box.

Figure 40 *– Overflow (**Source**: https://www.w3schools.com/css/css_overflow.asp)*

- hidden

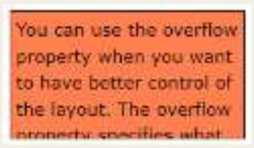

You can use the overflow property when you want to have better control of the layout. The overflow property specifies what

Figure 41 *– Overflow (**Source**: https://www.w3schools.com/css/css_overflow.asp)*

- Scroll

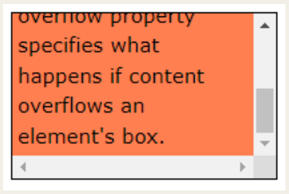

overflow property specifies what happens if content overflows an element's box.

***Figure 42** – Overflow (**Source**: https://www.w3schools.com/css/css_overflow.asp)*

- auto

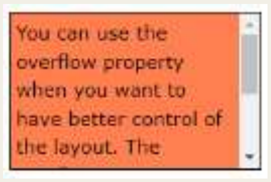

You can use the overflow property when you want to have better control of the layout. The

***Figure 43** – Overflow (**Source**: https://www.w3schools.com/css/css_overflow.asp)*

CSS Float

The float property is used for positioning and formatting content

Example - **float: right**

The following example specifies that an image should float to the **right** in a text:

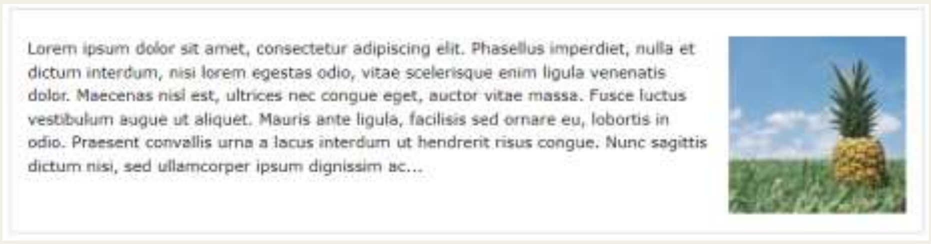
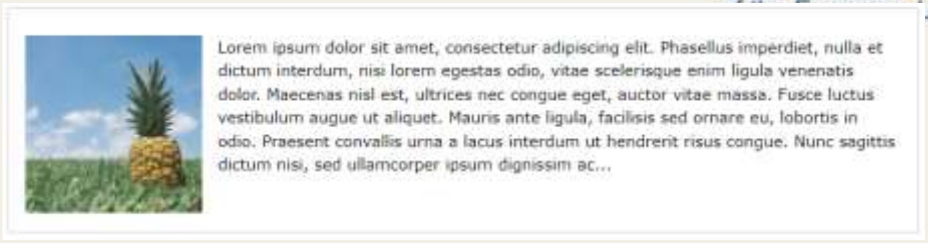

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

***Figure 44** – Float-right (**Source**: https://www.w3schools.com/css/css_float.asp)*

- Example - **float: left**

The following example specifies that an image should float to the **left** in a text:

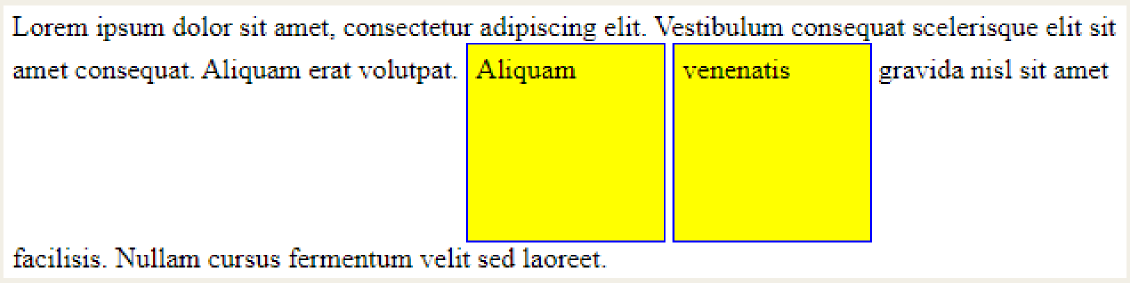


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Phasellus imperdiet, nulla et dictum interdum, nisi lorem egestas odio, vitae scelerisque enim ligula venenatis dolor. Maecenas nisl est, ultrices nec congue eget, auctor vitae massa. Fusce luctus vestibulum augue ut aliquet. Mauris ante ligula, facilisis sed ornare eu, lobortis in odio. Praesent convallis urna a lacus interdum ut hendrerit risus congue. Nunc sagittis dictum nisi, sed ullamcorper ipsum dignissim ac...

***Figure 45** – Float-left(**Source**: https://www.w3schools.com/css/css_float.asp)*

CSS Inline-block

- Display: inline-block allows to set a width and height on the element.
- With display: inline-block the top and bottom margins/paddings are respected
- Display: inline-block does not add a line-break after the element, so the element can sit next to other elements.



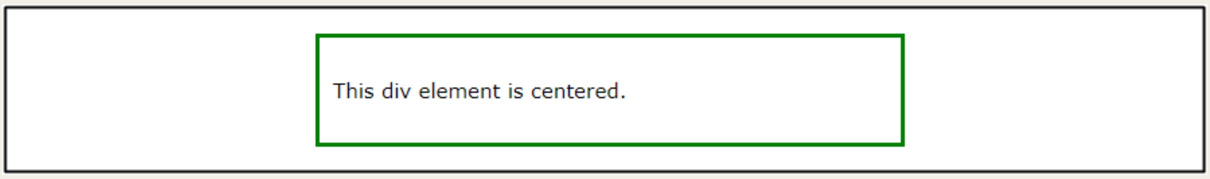
Lorem ipsum dolor sit amet, consectetur adipiscing elit. Vestibulum consequat scelerisque elit sit amet consequat. Aliquam erat volutpat. Aliquam venenatis gravida nisl sit amet facilisis. Nullam cursus fermentum velit sed laoreet.

***Figure 46** – Inline-block (**Source**: https://www.w3schools.com/css/css_inline-block.asp)*

CSS Align

- To horizontally center a block element (like <div>), use margin : auto
- Setting the width of the element will prevent it from stretching out to the edges of its container.

- The element will then take up the specified width, and the remaining space will be split equally between the two margins:



This div element is centered.

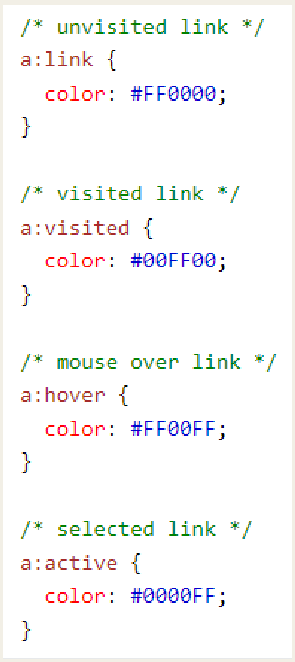
Figure 47 *– Align (**Source**: https://www.w3schools.com/css/css_align.asp)*

CSS Combinators

- 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) -> e.g. div p {background-color: yellow }
- child selector (>) -> e.g. div > p {background-color: yellow }
- adjacent sibling selector (+) -> e.g. div + p {background-color: yellow }
- general sibling selector (~) -> e.g. div ~ p {background-color: yellow }

CSS Pseudo-class

- 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

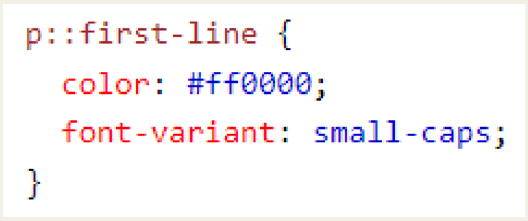


```
/* unvisited link */
a:link {
  color: #FF0000;
}

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

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

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

Figure 48 *– Pseudo Class (**Source**: https://www.w3schools.com/css/css_pseudo_classes.asp)*

CSS Pseudo-element

- 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



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

Figure 49 *– Pseudo element (**Source**: https://www.w3schools.com/css/css_pseudo_elements.asp)*

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

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

CSS Opacity

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

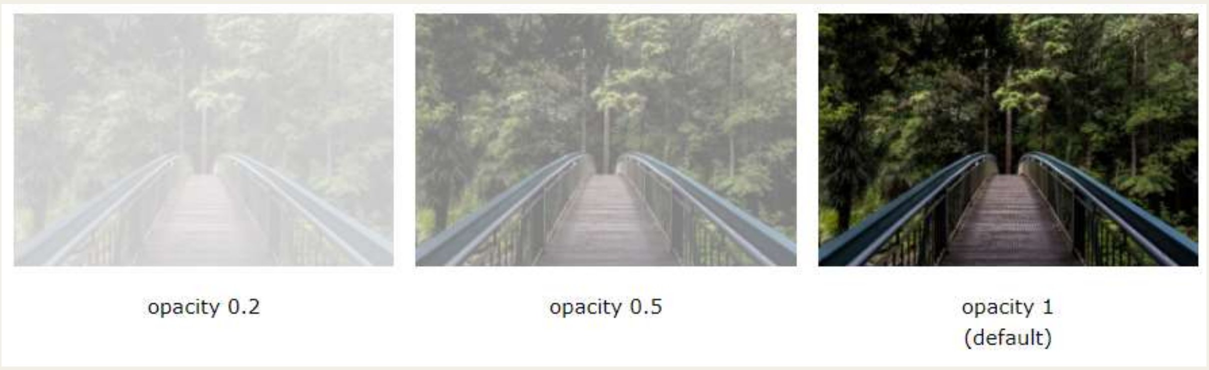


***Figure 50** – CSS Opacity (**Source**: https://www.w3schools.com/css/css_image_transparency.asp)*

CSS Navigation Bar

With CSS you can transform boring HTML menus into good-looking navigation bars.

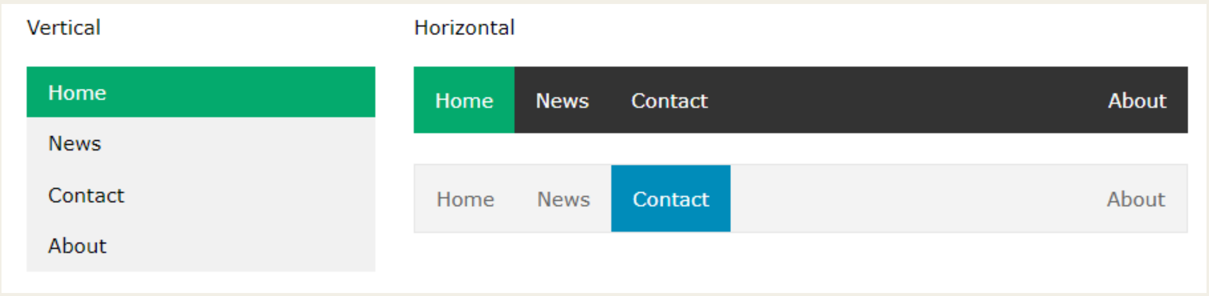


***Figure 51** – Navbar (**Source**: https://www.w3schools.com/css/css_navbar.asp)*

CSS Dropdowns

With CSS you can transform boring HTML menus into good-looking navigation bars.

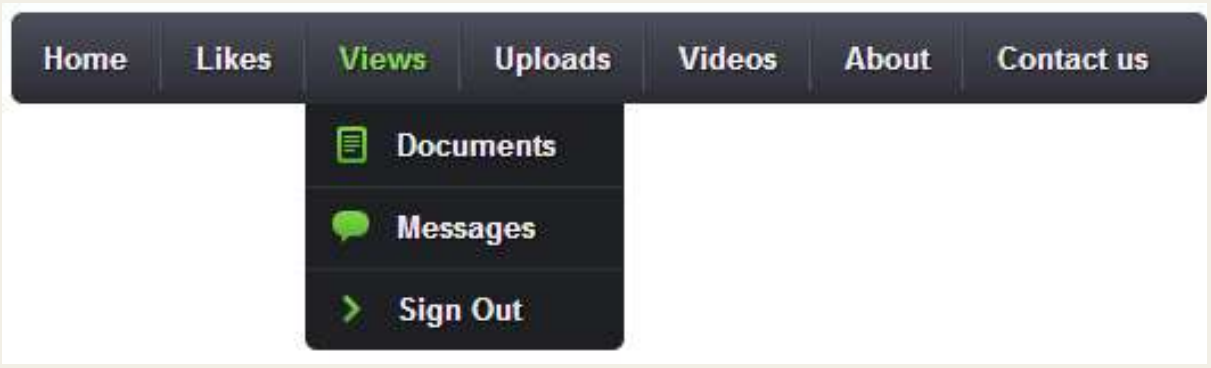


***Figure 52** – Dropdown (**Source**: <https://www.w3schools.com/css/css_dropdowns.asp>)*

CSS Image Gallery

CSS can be used to create an image gallery.

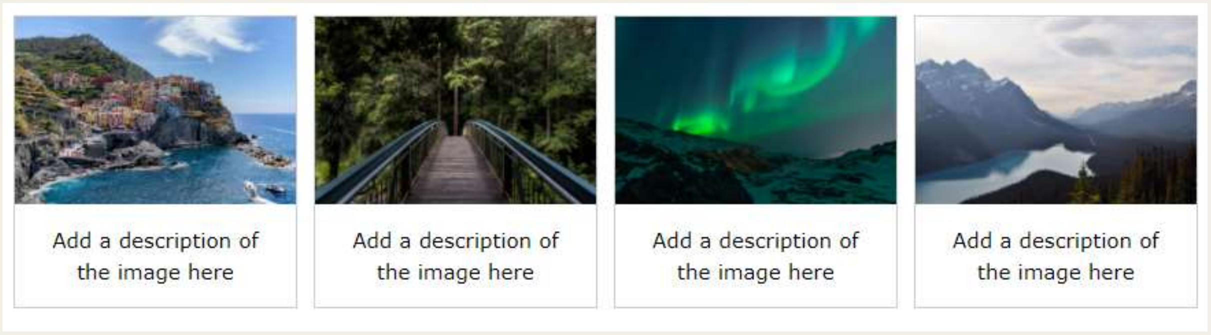

Add a description of the image here
Add a description of the image here
Add a description of the image here
Add a description of the image here

***Figure 53** – Image gallery (**Source**: <https://www.w3schools.com/css/css_image_gallery.asp>)*

CSS Image Sprites

- An image sprite is a collection of images put into a single image.
- A web page with many images can take a long time to load and generates multiple server requests.
- Using image sprites will reduce the number of server requests and save bandwidth.

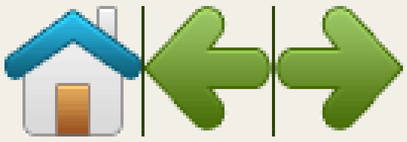
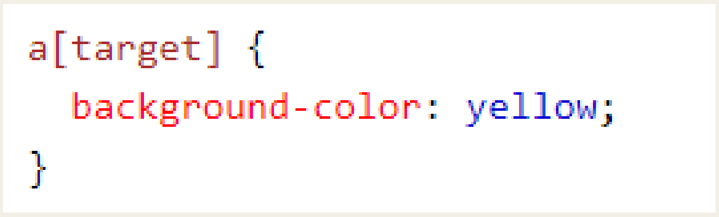


Figure 54 *– Image sprites (**Source**: https://www.w3schools.com/css/css_image_sprites.asp)*

CSS Attr Selectors

- The [attribute] selector is used to select elements with a specified attribute.
- The following example selects all <a> elements with a target attribute:



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

Figure 55 *– Attr Selector (**Source**: https://www.w3schools.com/css/css_attribute_selectors.asp)*

CSS Forms

The look of an HTML form can be greatly improved with CSS:

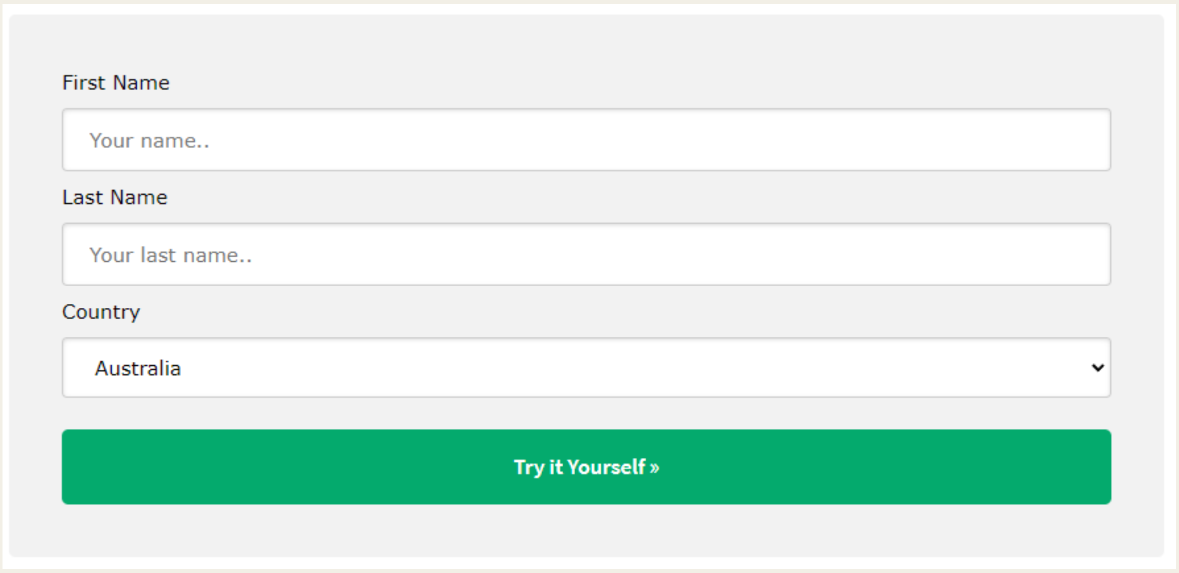


Figure 56 – *Forms (Source: https://www.w3schools.com/css/css_form.asp)*

CSS Counters

CSS counters are "variables" maintained by CSS whose values can be incremented by CSS rules (to track how many times they are used). Counters let you adjust the appearance of content based on its placement in the document.

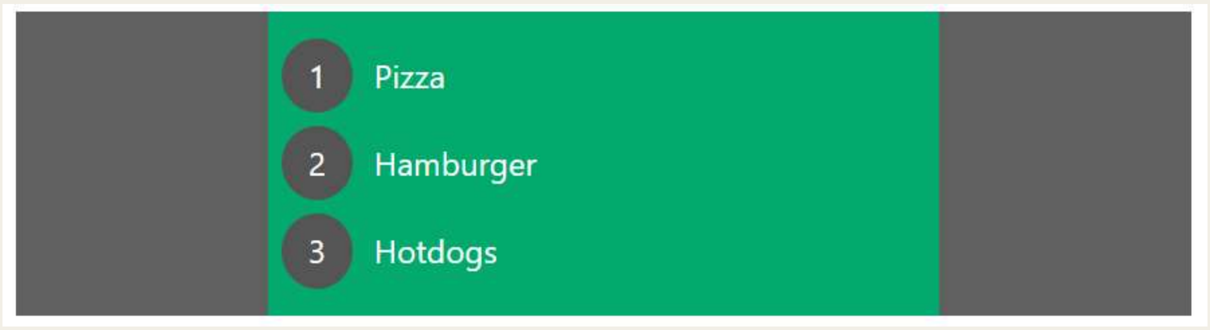


Figure 57 – *Counter (Source: https://www.w3schools.com/css/css_counters.asp)*

CSS Website Layout

A website is often divided into headers, menus, content and a footer:

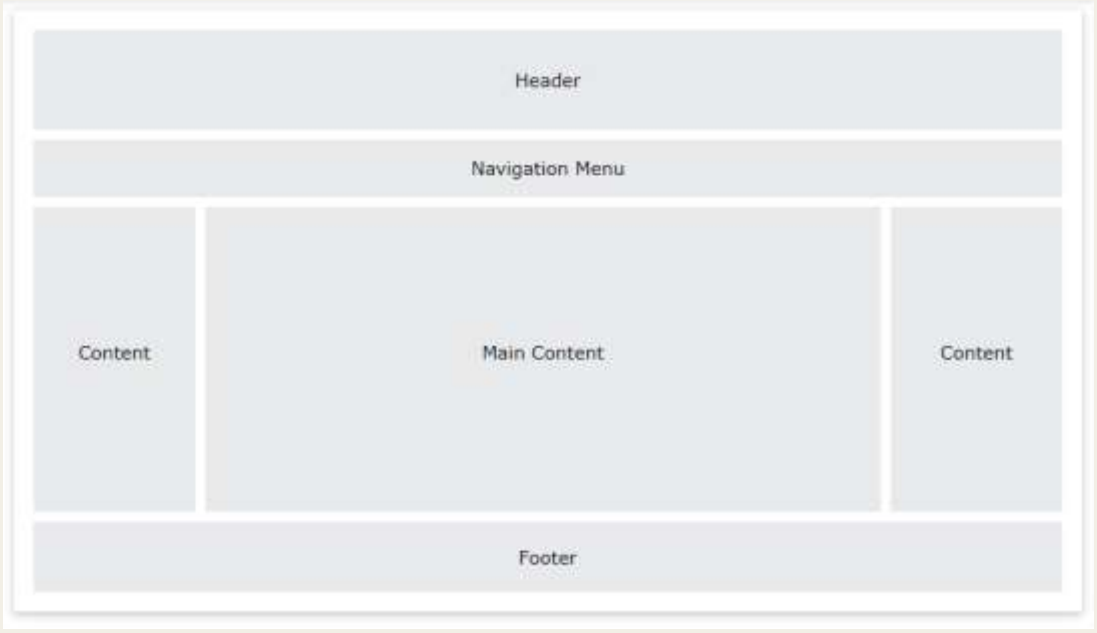
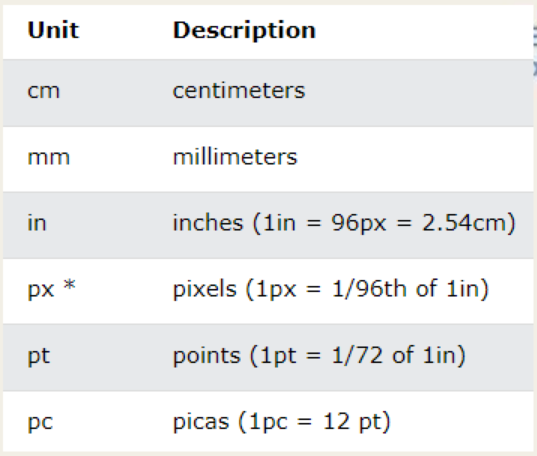


Figure 58 *– Website layout (**Source**: https://www.w3schools.com/css/css_website_layout.asp)*

CSS Units

- CSS has several different units for expressing a length.
- Many CSS properties take "length" values, such as width, margin, padding, font-size, etc,
- Length is a number followed by a length unit, such as 10px, 2em, etc,

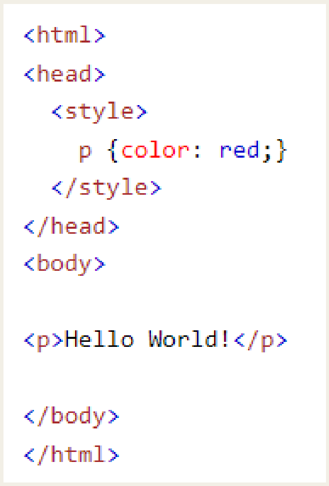


Unit	Description
cm	centimeters
mm	millimeters
in	inches (1in = 96px = 2.54cm)
px *	pixels (1px = 1/96th of 1in)
pt	points (1pt = 1/72 of 1in)
pc	picas (1pc = 12 pt)

Figure 59 *– Units (**Source**: https://www.w3schools.com/css/css_units.asp)*

CSS 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 are ultimately applied to an element.



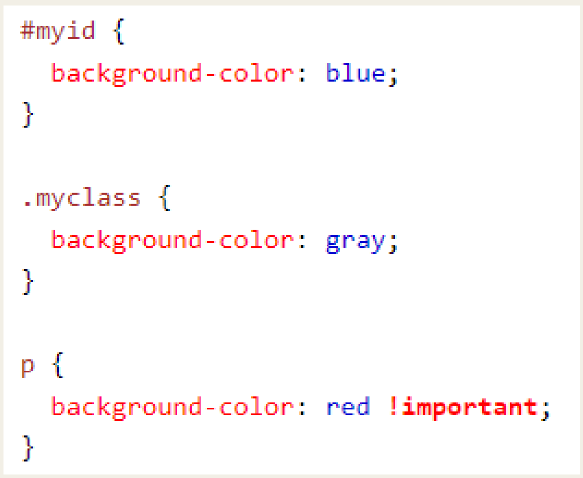
```
<html>
<head>
  <style>
    p {color: red;}
  </style>
</head>
<body>

<p>Hello World!</p>

</body>
</html>
```

Figure 60 *– Specificity (**Source**: https://www.w3schools.com/css/css_specificity.asp)*

In this example, we have used the "p" element as selector and specified a red color for this element. The text will be red.

CSS !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!

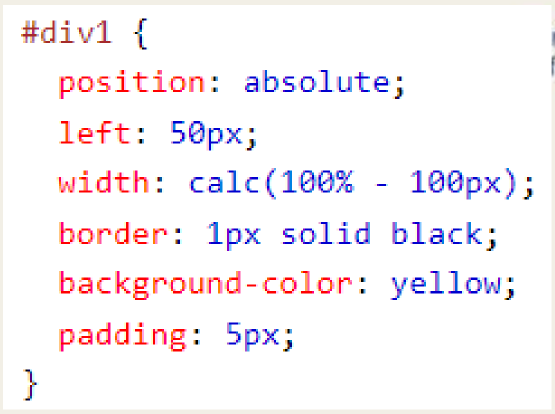


```
#myid {
  background-color: blue;
}

.myclass {
  background-color: gray;
}

p {
  background-color: red !important;
}
```

***Figure 61** – !important (**Source**: https://www.w3schools.com/css/css_important.asp)*

CSS Math Function

- The CSS math functions allow mathematical expressions to be used as property values. Some Math functions are: calc(), max() and min() functions.
- Example: the use of calc() to calculate the width of a <div> element:



```
#div1 {
  position: absolute;
  left: 50px;
  width: calc(100% - 100px);
  border: 1px solid black;
  background-color: yellow;
  padding: 5px;
}
```

***Figure 62** – Math function (**Source**: https://www.w3schools.com/css/css_math_functions.asp)*

Advanced CSS

CSS rounded corners

The border-radius property defines the radius of an element's corners.

Tip: This property allows you to add rounded corners to elements!

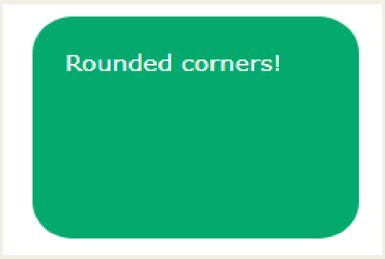


***Figure 63** – Rounded corners (**Source**: https://www.w3schools.com/css/css3_borders.asp)*

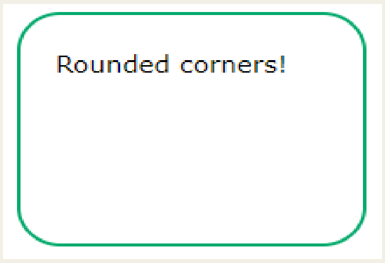


***Figure 64** – Rounded corners (**Source**: https://www.w3schools.com/css/css3_borders.asp)*

CSS Border Images

The CSS border-image property allows you to specify an image to be used instead of the normal border around an element.

The property has three parts:

- The image to use as the border
- Where to slice the image
- Define whether the middle sections should be repeated or stretched

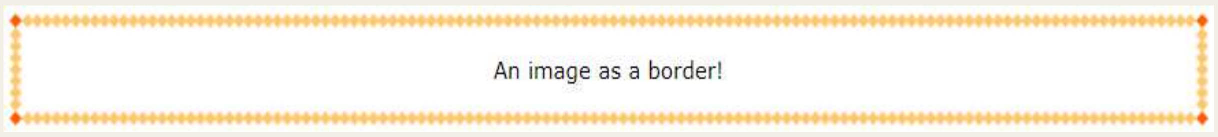
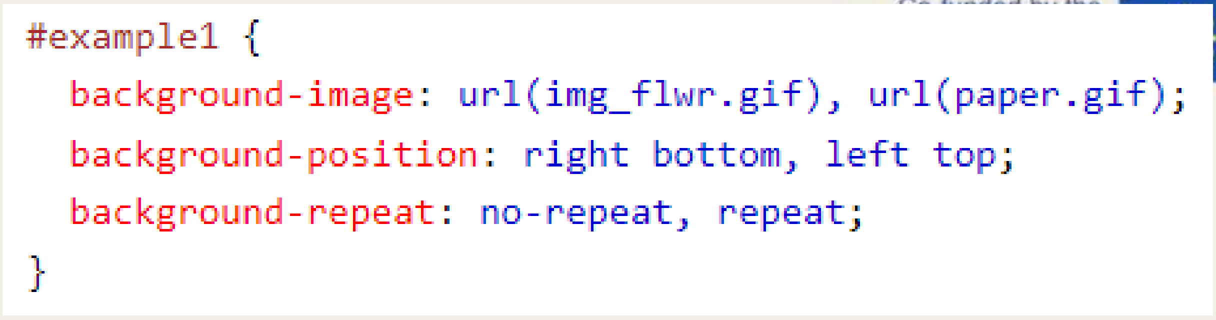


Figure 65 *– Border image (**Source**: https://www.w3schools.com/css/css3_border_images.asp)*

CSS Backgrounds

CSS allows you to add multiple background images for an element, through the background-image property.

The following example has two background images, the first image is a flower (aligned to the bottom and right) and the second image is a paper background (aligned to the top-left corner)



```
#example1 {
  background-image: url(img_flwr.gif), url(paper.gif);
  background-position: right bottom, left top;
  background-repeat: no-repeat, repeat;
}
```

***Figure 66** – Backgrounds (**Source**: https://www.w3schools.com/css/css3_backgrounds.asp)*

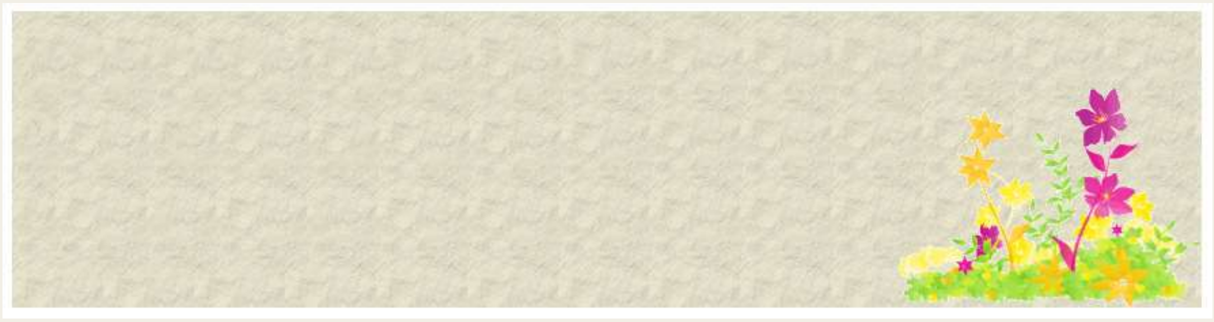
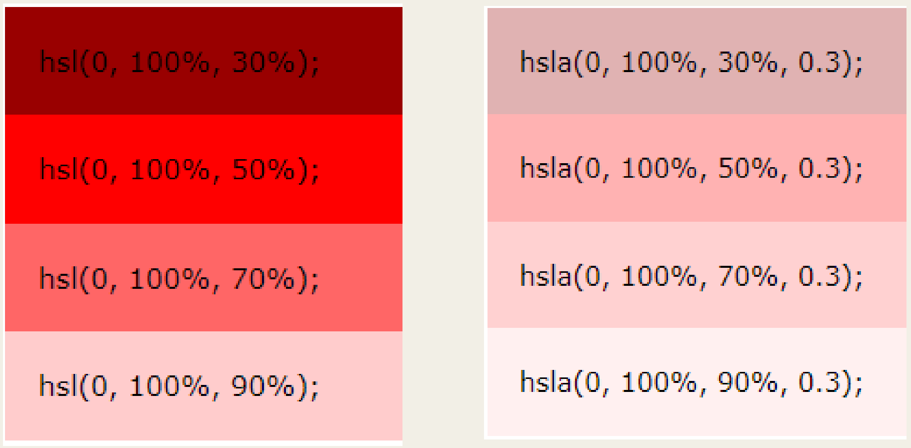


***Figure 67** – Backgrounds (**Source**: https://www.w3schools.com/css/css3_backgrounds.asp)*

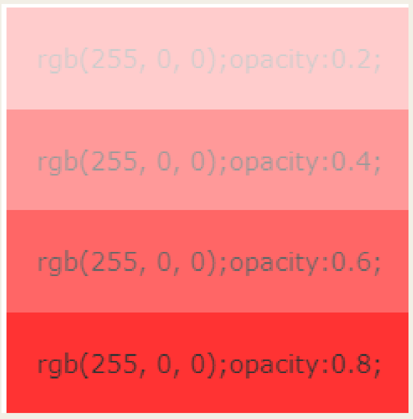
CSS Colors

- CSS supports +140 color names, HEX values, RGB values, RGBA values, HSL values, HSLA values, and opacity.

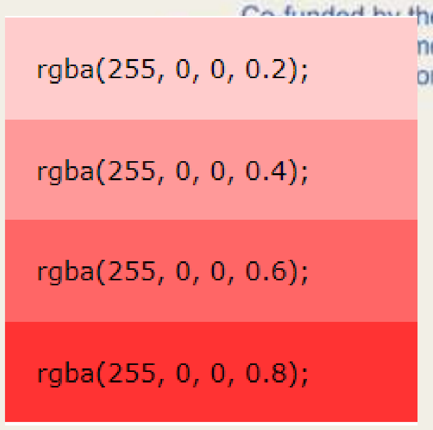


hsl(0, 100%, 30%);	hsla(0, 100%, 30%, 0.3);
hsl(0, 100%, 50%);	hsla(0, 100%, 50%, 0.3);
hsl(0, 100%, 70%);	hsla(0, 100%, 70%, 0.3);
hsl(0, 100%, 90%);	hsla(0, 100%, 90%, 0.3);

***Figures 68 and 69** – CSS Colors (**Source**: https://www.w3schools.com/css/css3_colors.asp)*



rgb(255, 0, 0);opacity:0.2;
rgb(255, 0, 0);opacity:0.4;
rgb(255, 0, 0);opacity:0.6;
rgb(255, 0, 0);opacity:0.8;



rgba(255, 0, 0, 0.2);
rgba(255, 0, 0, 0.4);
rgba(255, 0, 0, 0.6);
rgba(255, 0, 0, 0.8);

Figures 70 and 71 *– CSS Colors (**Source**: https://www.w3schools.com/css/css3_colors.asp)*

CSS Color Keywords

This page will explain the transparent, currentcolor, and inherit keywords:

- The transparent keyword is used to make a color transparent. This is often used to make a transparent background color for an element.
- The currentcolor keyword is like a variable that holds the current value of the color property of an element.

This keyword can be useful if you want a specific color to be consistent in an element or a page.

The inherit keyword specifies that a property should inherit its value from its parent element.

The inherit keyword can be used for any CSS property, and on any HTML element.

CSS Gradient

CSS gradients let you display smooth transitions between two or more specified colors.

CSS defines three types of gradients:

- Linear Gradients (goes down/up/left/right/diagonally)
- Radial Gradients (defined by their center)
- Conic Gradients (rotated around a center point)

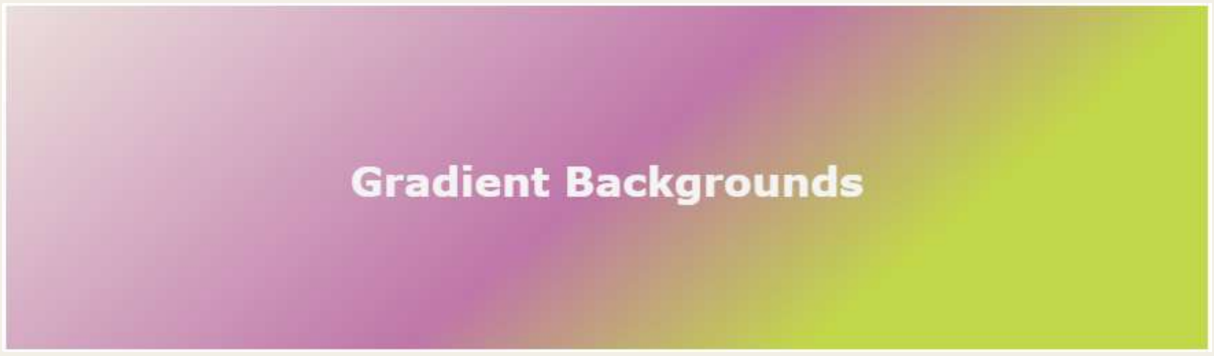


Figure 72 *– Gradient Backgrounds (**Source**: https://www.w3schools.com/css/css3_gradients.asp)*

CSS Shadows

With CSS you can add shadow to text and to elements.

In these chapters you will learn about the following properties :

- Text-shadow
- Box-shadow

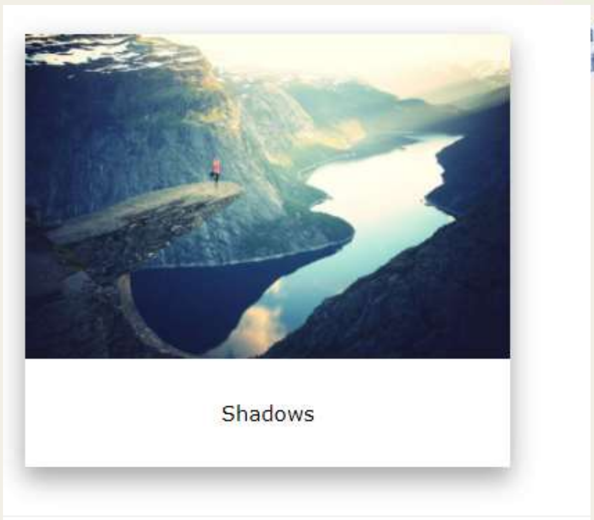


***Figure 73** – Shadows (**Source**: https://www.w3schools.com/css/css3_shadows.asp)*

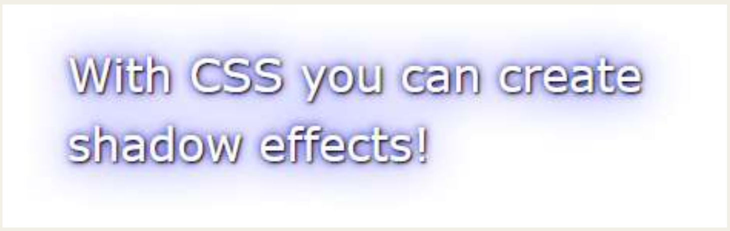


***Figure 74** – Shadows (**Source**: https://www.w3schools.com/css/css3_shadows.asp)*

CSS Text Effects

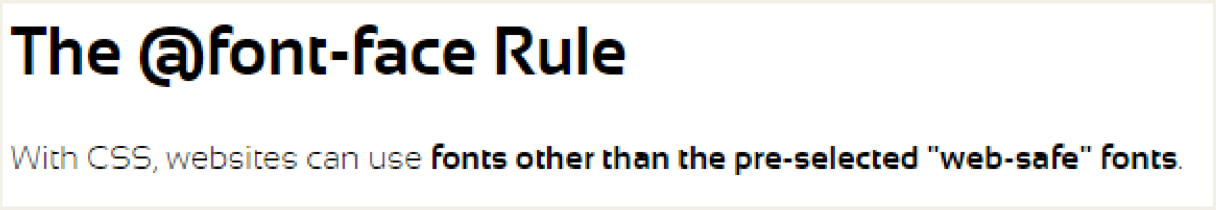
In this chapter you will learn about Text-overflow, word-wrap, word-break, writing-mode:

- The text-overflow property specifies how overflowed content that is not displayed should be signaled to the user.
- The CSS word-wrap property allows long words to be able to be broken and wrap onto the next line.
- The CSS word-break property specifies line breaking rules.

- The CSS writing-mode property specifies whether lines of text are laid out horizontally or vertically.

CSS Web Fonts

- Web fonts allow Web designers to use fonts that are not installed on the user's computer.
- When you have found/bought the font you wish to use, just include the font file on your web server, and it will be automatically downloaded to the user when needed.
- Your "own" fonts are defined within the CSS @font-face rule



The @font-face Rule

With CSS, websites can use **fonts other than the pre-selected "web-safe" fonts**.

Figure 75 *– Web fonts (**Source**: https://www.w3schools.com/css/css3_fonts.asp)*

CSS 2D transforms

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

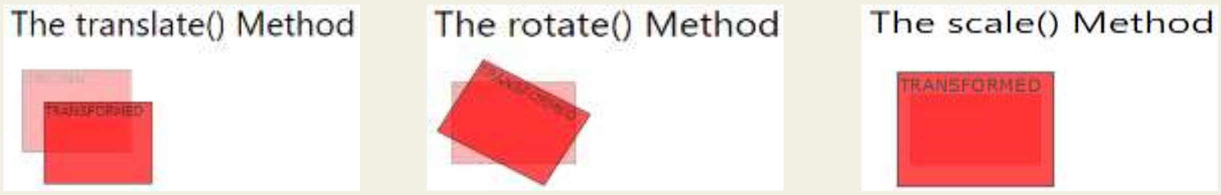


Figure 76 *– 2D Transform (**Source**: https://www.w3schools.com/css/css3_2dtransforms.asp)*

CSS 3D transforms

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

- RotateX()
- RotateY()
- RotateZ()

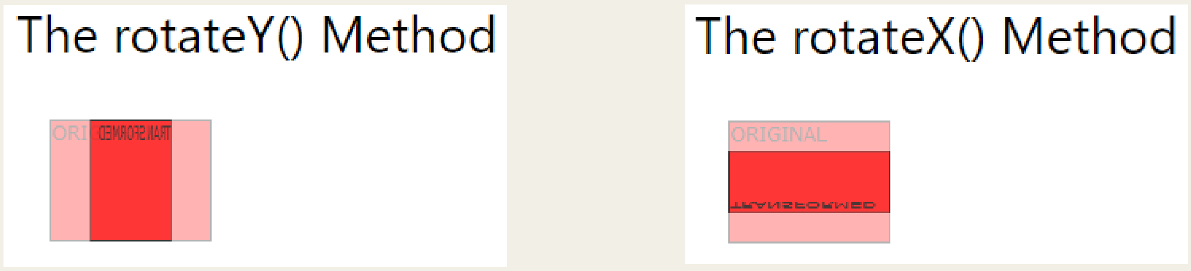
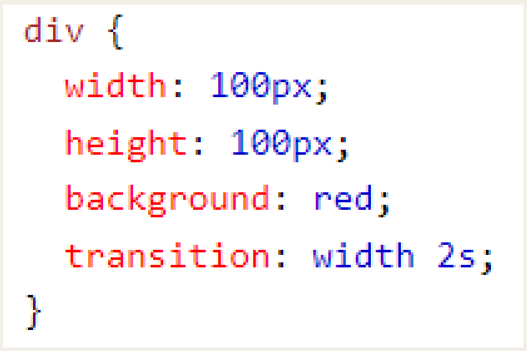


***Figure 78** – 3D Transform (**Source**: https://www.w3schools.com/css/css3_3dtransforms.asp)*

CSS Transitions

- CSS transitions allows you to change property values smoothly, over a given duration.
- To create a transition effect, you must specify two things:
  - the CSS property you want to add an effect to
  - the duration of the effect
- The following example shows a 100px * 100px red <div> element. The <div> element has also specified a transition effect for the width property, with a duration of 2 seconds:

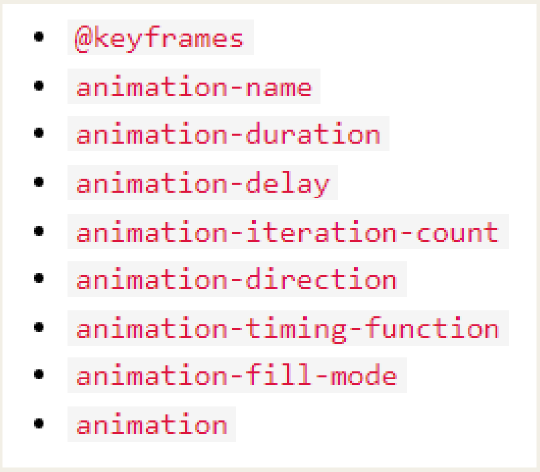


```
div {
  width: 100px;
  height: 100px;
  background: red;
  transition: width 2s;
}
```

***Figure 79** – Transitions (**Source**: https://www.w3schools.com/css/css3_transitions.asp)*

CSS Animations

- An animation lets an element gradually change from one style to another.
- You can change as many CSS properties you want, as many times as you want.
- To use CSS animation, you must first specify some keyframes for the animation.
- Keyframes hold what styles the element will have at certain times.
- These are the main the main animation's properties :



- `@keyframes`
- `animation-name`
- `animation-duration`
- `animation-delay`
- `animation-iteration-count`
- `animation-direction`
- `animation-timing-function`
- `animation-fill-mode`
- `animation`

***Figure 80** – Animations (**Source**: https://www.w3schools.com/css/css3_animations.asp)*

Co-funded by the Erasmus+ Programme of the European Union

CSS Tooltips

- A tooltip is often used to specify extra information about something when the user moves the mouse pointer over an element:

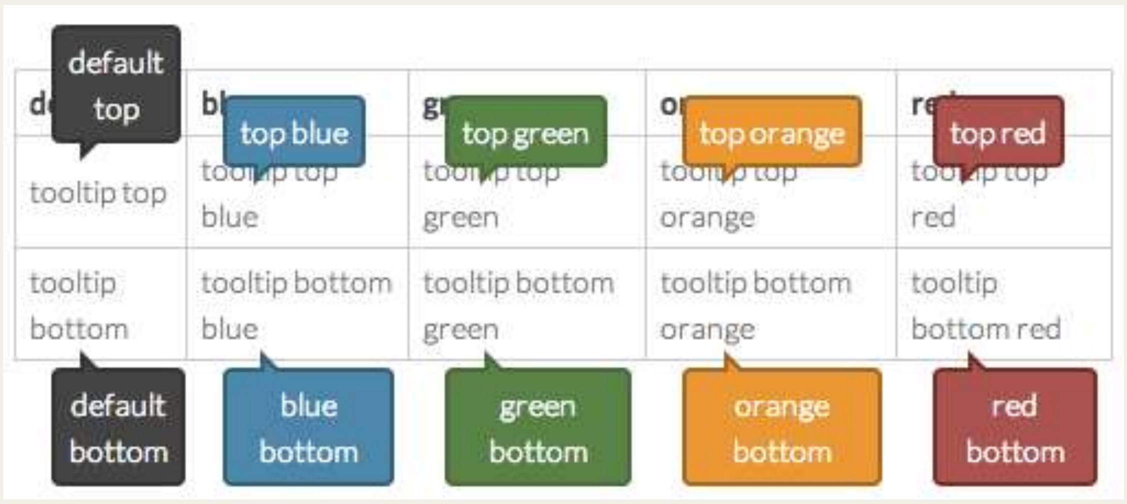


***Figure 81** – Tooltip*

CSS Style images

- Using CSS to style images allows you to uniformly specify how images should appear across your website with only a few rulesets :

  Add a border, change the shape and size of the image, …

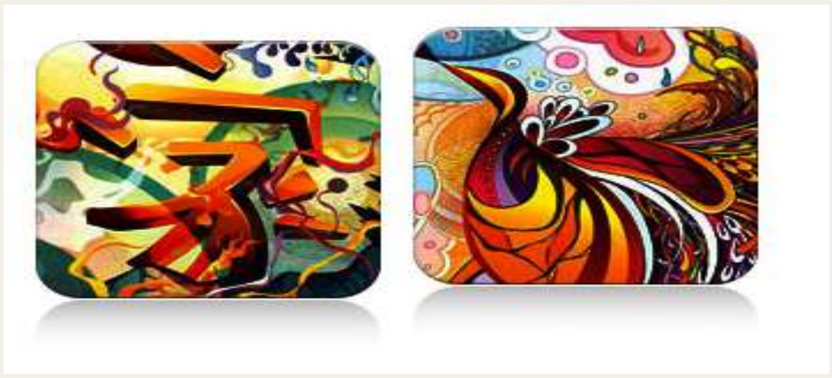


Figure 82 *– Images*

CSS image reflection

- The box-reflect property is used to create an image reflection.
- The value of the The box-reflect property can be : below, above, left, or right

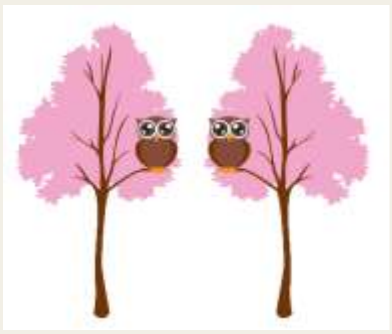


Figure 83 *– Image reflection (**Source**: https://www.w3schools.com/css/css3_image_reflection.asp)*

CSS Object-fit

The CSS object-fit property is used to specify how an <img> or <video> should be resized to fit its container.

This property tells the content to fill the container in a variety of ways; such as "preserve that aspect ratio" or "stretch up and take up as much space as possible".

Look at the following image from Paris. This image is 400 pixels wide and 300 pixels high:

Figure 84 *– Object fit (**Source**: https://www.w3schools.com/css/css3_object-fit.asp)*

CSS Object-position

Let's say that the part of the image that is shown, is not positioned as we want. To position the image, we will use the object-position property

Here we will use the object-position property to position the image so that the great old building is in center:

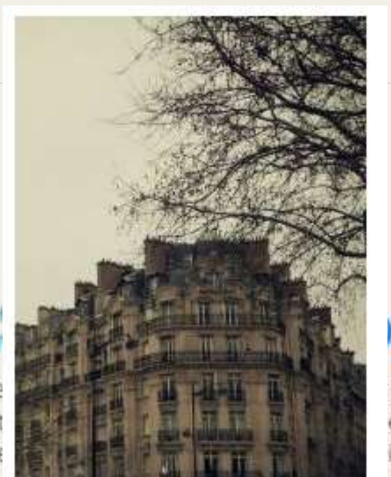


*Figure 85 – Object position (**Source**: https://www.w3schools.com/css/css3_object-position.asp)*

CSS Masking

- With CSS masking you create a mask layer to place over an element to hide portions of the element partially or fully.
- The CSS mask-image property specifies a mask layer image.

  The mask layer image can be a PNG image, an SVG image, a CSS gradient, or an SVG <mask>element.

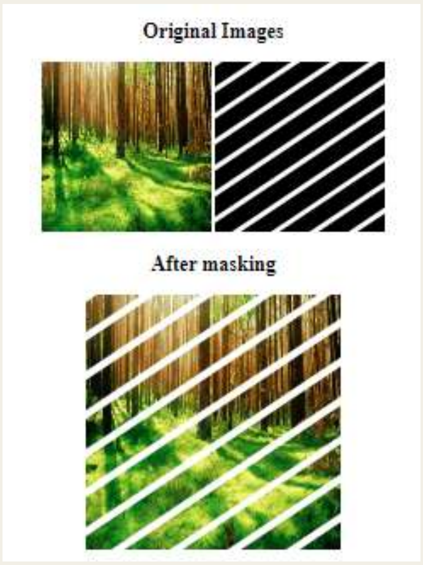


***Figure 86** – Masking (**Source**: https://www.w3schools.com/css/css3_masking.asp)*

CSS Buttons

- There are lots of properties to style a button in CSS
- This is an example:

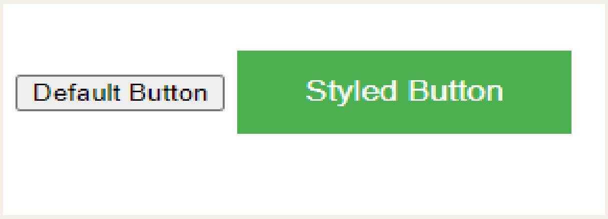
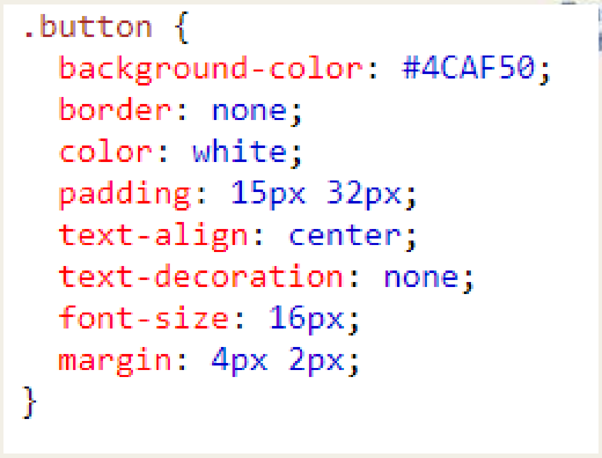


***Figure 87** – Button (**Source**: https://www.w3schools.com/css/css3_buttons.asp)*



```
.button {
  background-color: #4CAF50;
  border: none;
  color: white;
  padding: 15px 32px;
  text-align: center;
  text-decoration: none;
  font-size: 16px;
  margin: 4px 2px;
}
```

***Figure 88** – Button (**Source**: https://www.w3schools.com/css/css3_buttons.asp)*

CSS Pagination

If you have a website with lots of pages, you may wish to add some sort of pagination to each page and these are some examples:

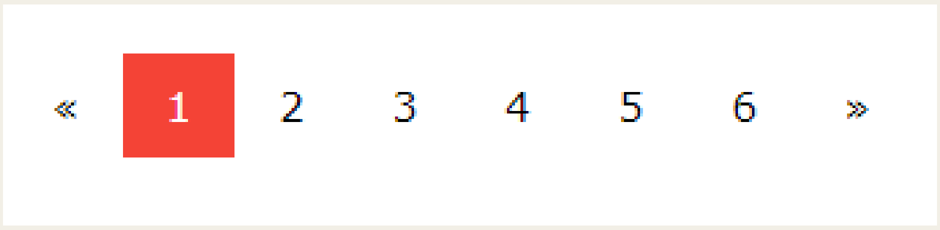


***Figure 90** – Pagination (**Source**: https://www.w3schools.com/css/css3_pagination.asp)*

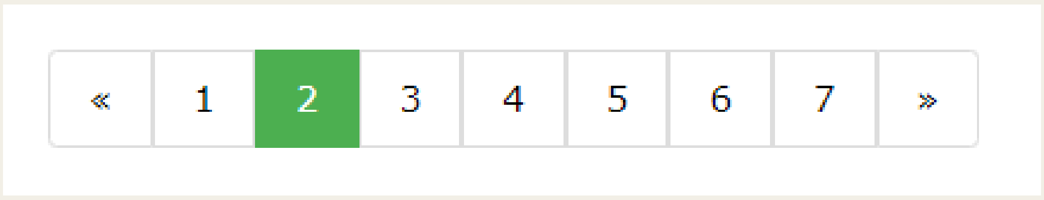
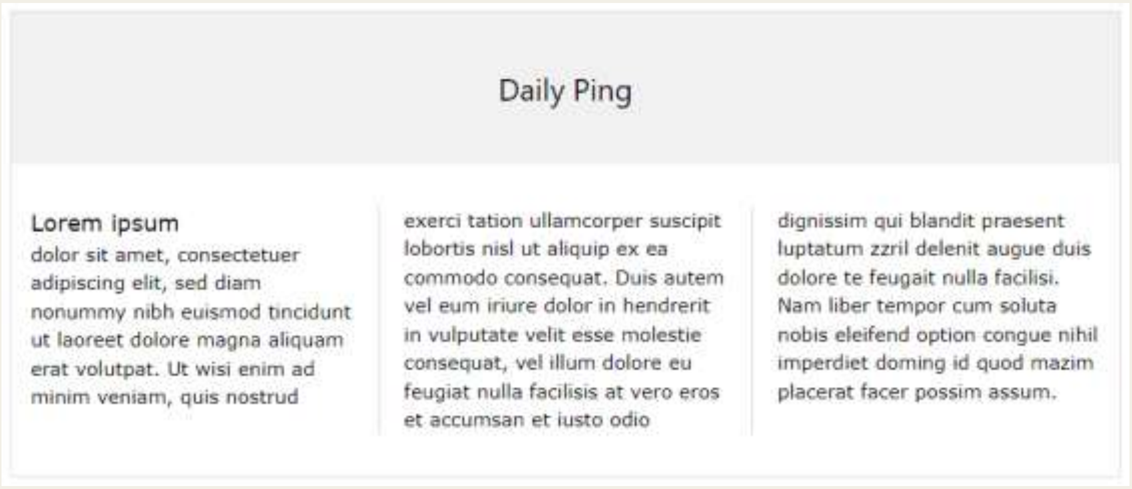


***Figure 91** – Pagination (**Source**: https://www.w3schools.com/css/css3_pagination.asp)*

CSS Multiple Columns

The CSS multi-column layout allows easy definition of multiple columns of text - just like in newspapers:


Daily Ping

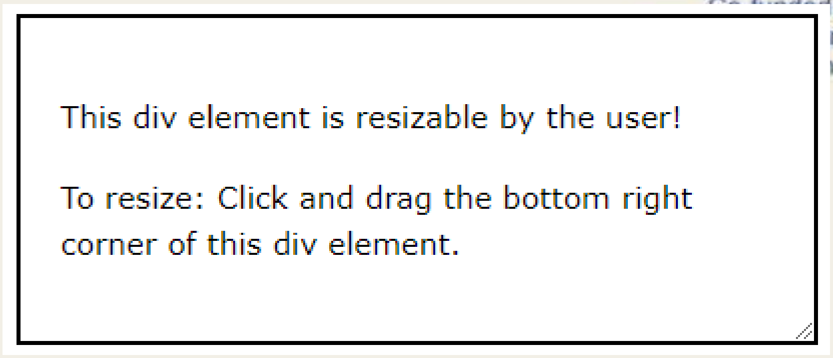
Lorem ipsum
dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

***Figure 92** – Multiple Columns (**Source**: https://www.w3schools.com/css/css3_multiple_columns.asp)*

CSS User Interface

In this chapter you will learn about the following CSS user interface properties:

- The resize property specifies if (and how) an element should be resizable by the user.


This div element is resizable by the user!

To resize: Click and drag the bottom right corner of this div element.

Figure 93 *– UI (**Source**: https://www.w3schools.com/css/css3_user_interface.asp)*

- The outline-offset property adds space between an outline and the edge or border of an element.

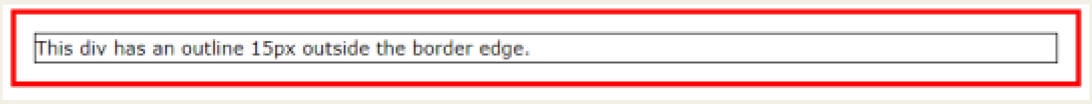

This div has an outline 15px outside the border edge.

Figure 94 *– UI (**Source**: https://www.w3schools.com/css/css3_user_interface.asp)*

CSS Variables

- The var() function is used to insert the value of a CSS variable.
- CSS variables have access to the DOM, which means that you can create variables with local or global scope, change the variables with JavnceaScript, and change the variables based on media queries.
- A good way to use CSS variables is when it comes to the colors of your design. Instead of copy and paste the same colors over and over again, you can place them in variables.

CSS Box sizing

- The box-sizing property in CSS defines how the user should calculate the total width and height of an element i.e padding and borders, are to be included or not.

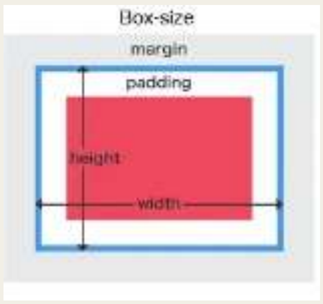


Figure 95 *– Box sizing (**Source**: https://www.w3schools.com/css/css3_box-sizing.asp)*

CSS Media Queries

Media queries in CSS3 extended the CSS2 media types idea: Instead of looking for a type of device, they look at the capability of the device.
Media queries can be used to check many things, such as:

- width and height of the viewport
- width and height of the device
- orientation (is the tablet/phone in landscape or portrait mode?)
- resolution

Using media queries are a popular technique for delivering a tailored style sheet to desktops, laptops, tablets, and mobile phones (such as iPhone and Android phones).

CSS MQ examples

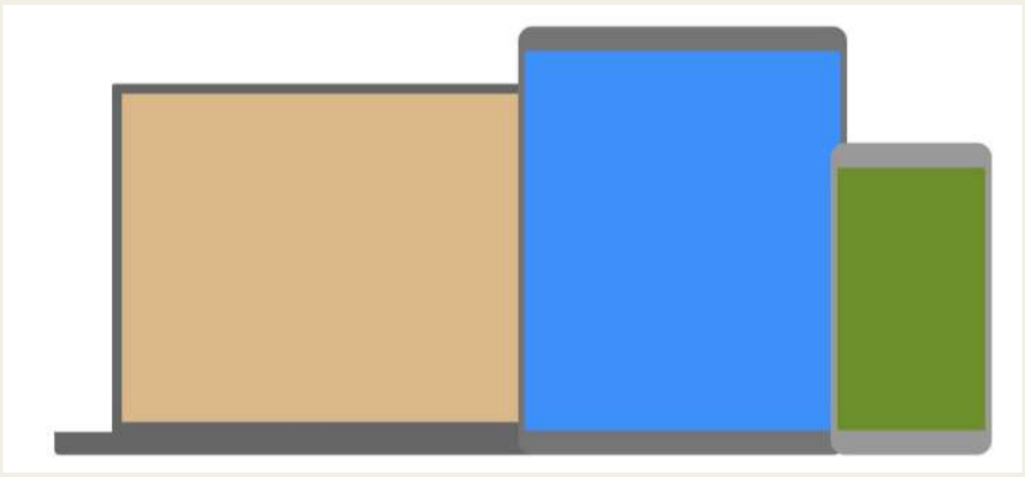


Figure 96 *– MQ Example (**Source**: https://www.w3schools.com/css/css3_mediaqueries_ex.asp)*

CSS Flexbox

The Flexible Box Layout Module makes it easier to design flexible responsive layout structure without using float or positioning.

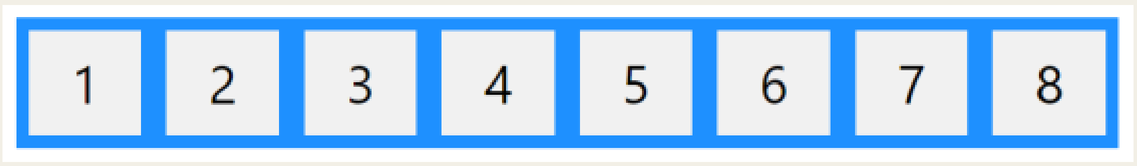


Figure 97 *– Flexbox (**Source**: https://www.w3schools.com/css/css3_flexbox.asp)*

Responsive Web Design (RWD)

RWD Intro

Responsive web design makes your web page look good on all devices.

Responsive web design uses only HTML and CSS.

Web pages can be viewed using many different devices: desktops, tablets, and phones. Your web page should look good, and be easy to use, regardless of the device.

Web pages should not leave out information to fit smaller devices, but rather adapt its content to fit any device:

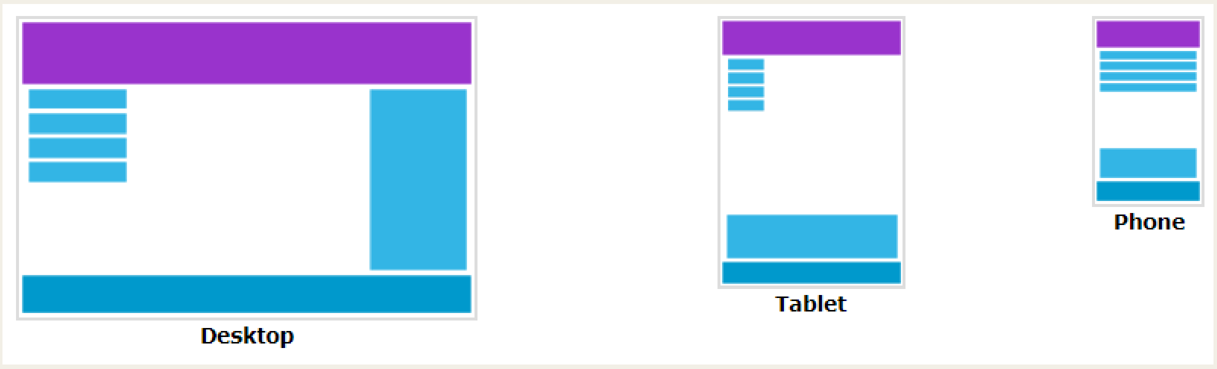


Figure 98 *– RWD (**Source**: https://www.w3schools.com/css/css_rwd_intro.asp)*

It is called responsive web design when you use CSS and HTML to resize, hide, shrink, enlarge, or move the content to make it look good on any screen.

RWD Viewport

The viewport is the user's visible area of a web page.

The viewport varies with the device and will be smaller on a mobile phone than on a computer screen.

Before tablets and mobile phones, web pages were designed only for computer screens, and it was common for web pages to have a static design and a fixed size.

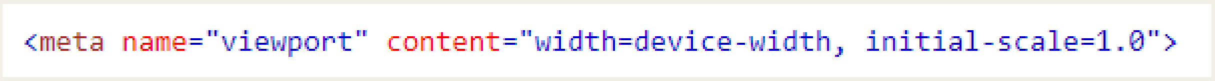
Then, when we started surfing the internet using tablets and mobile phones, fixed size web pages were too large to fit the viewport. To fix this, browsers on those devices scaled down the entire web page to fit the screen.

This was not perfect!! But a quick fix.

Setting the Viewport

HTML5 introduced a method to let web designers take control over the viewport, through the <meta> tag.

You should include the following <meta> viewport element in all your web pages:



```
<meta name="viewport" content="width=device-width, initial-scale=1.0">
```

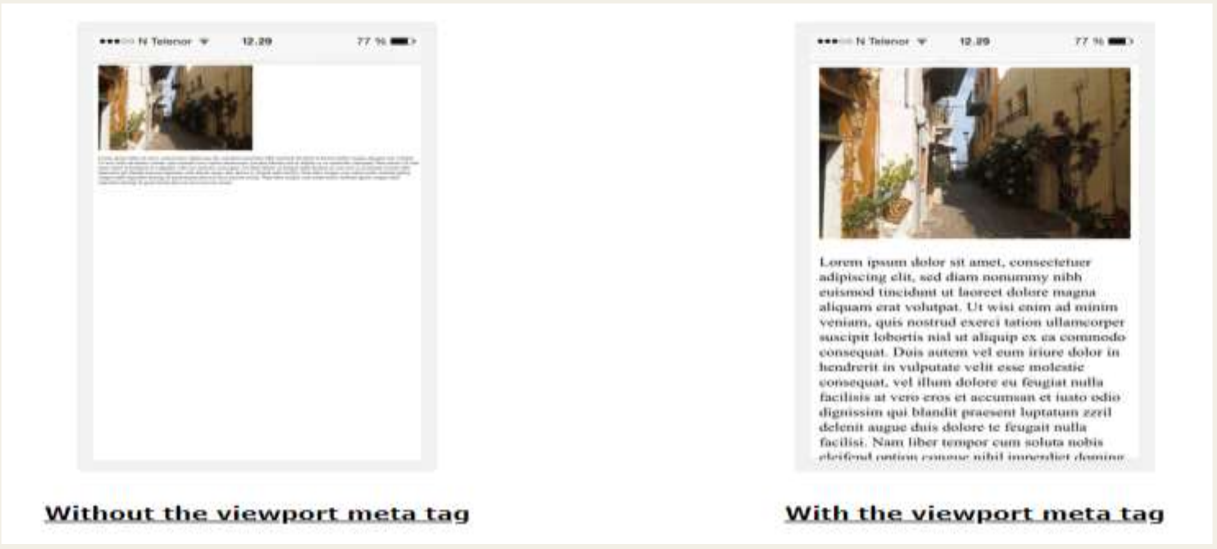
***Figure 99** – Setting Viewport (**Source**: https://www.w3schools.com/css/css_rwd_viewport.asp)*

This gives the browser instructions on how to control the page's dimensions and scaling.

The width=device-width part sets the width of the page to follow the screen-width of the device (which will vary depending on the device).

The initial-scale=1.0 part sets the initial zoom level when the page is first loaded by the browser.

Here is an example of a web page without the viewport meta tag, and the same web page with the viewport meta tag:



Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming

Figure 100 *– Viewport (**Source**: https://www.w3schools.com/css/css_rwd_viewport.asp)*

RWD Grid-View

Many web pages are based on a grid-view, which means that the page is divided into columns:

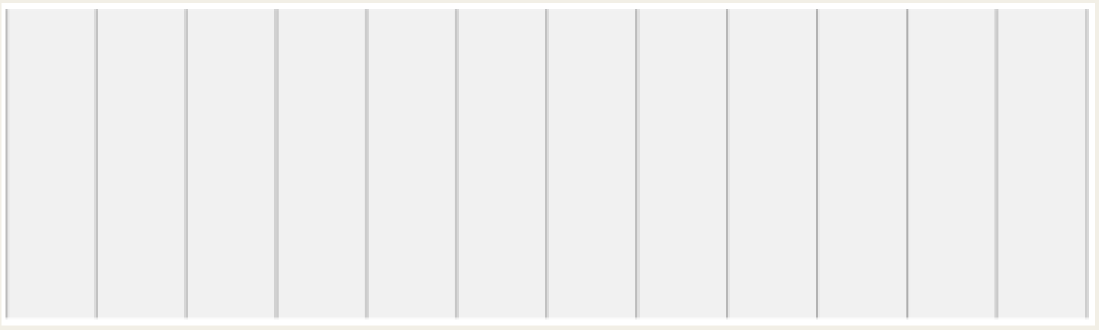


Figure 101 *– Grid-View (**Source**: https://www.w3schools.com/css/css_rwd_grid.asp)*

Many web pages are based on a grid-view, which means that the page is divided into columns:

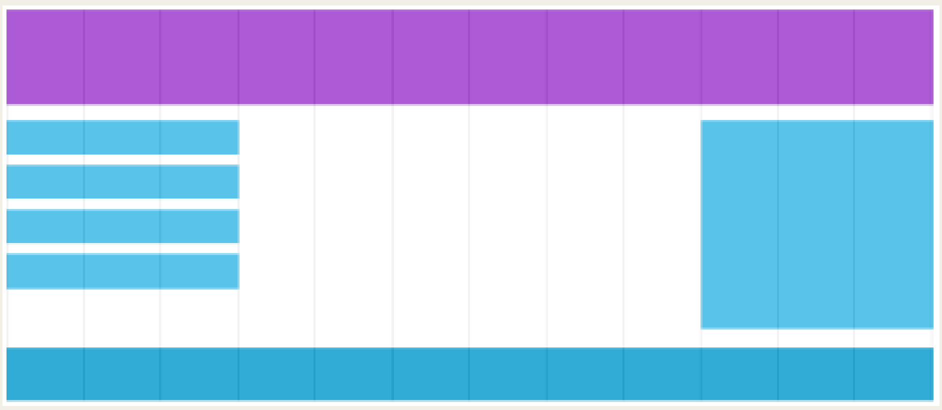
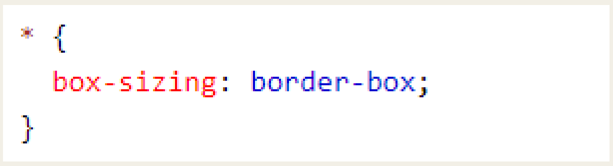


***Figure 102** – Grid-View (**Source**: https://www.w3schools.com/css/css_rwd_grid.asp)*

Building a responsive Grid-View

First ensure that all HTML elements have the box-sizing property set to border-box. This makes sure that the padding and border are included in the total width and height of the elements.

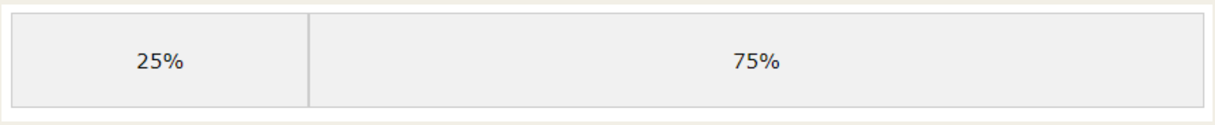
Add the following code in your CSS:



```
* {
  box-sizing: border-box;
}
```

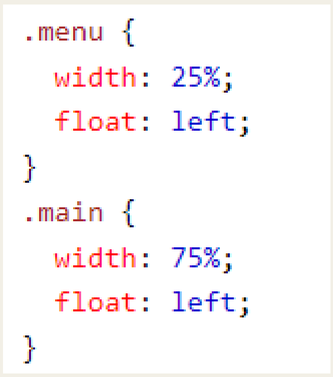
***Figure 103** – Responsive Grid-View (**Source**: https://www.w3schools.com/css/css_rwd_grid.asp)*

The following example shows a simple responsive web page, with two columns:



25%	75%

***Figure 104** – Responsive Grid-View (**Source**: https://www.w3schools.com/css/css_rwd_grid.asp)*

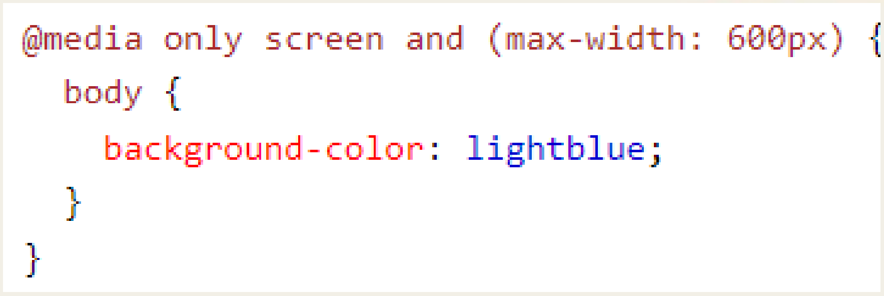


```
.menu {
  width: 25%;
  float: left;
}
.main {
  width: 75%;
  float: left;
}
```

***Figure 105** – Responsive Grid-View (**Source**: https://www.w3schools.com/css/css_rwd_grid.asp)*

RWD Media Queries

- Media query is a CSS technique introduced in CSS3.
- It uses the @media rule to include a block of CSS properties only if a certain condition is true.
- If the browser window is 600px or smaller, the background color will be lightblue:



```
@media only screen and (max-width: 600px) {
  body {
    background-color: lightblue;
  }
}
```

Figure 106 – *Media query (**Source**: https://www.w3schools.com/css/css_rwd_mediaqueries.asp)*

Add a breakpoint

Media queries can help with that. We can add a breakpoint where certain parts of the design will behave differently on each side of the breakpoint.

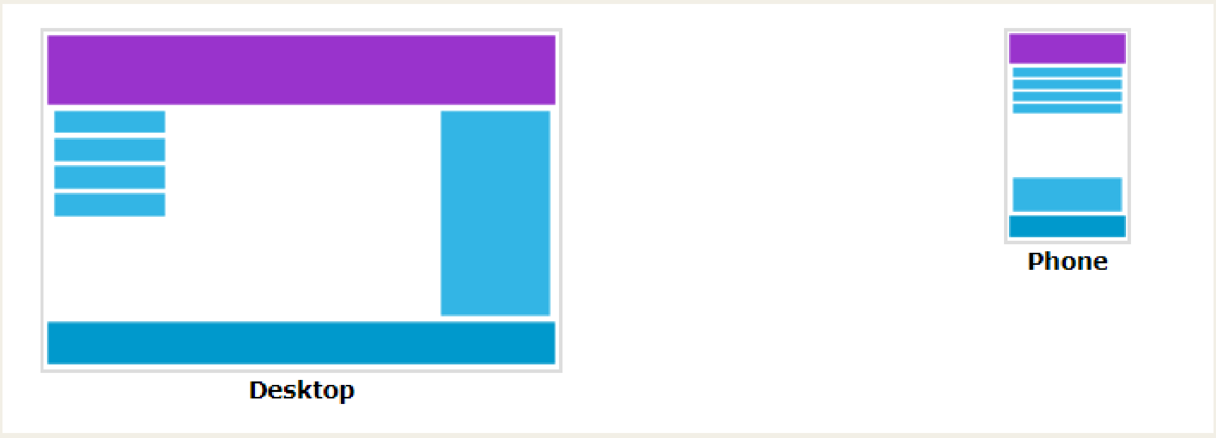
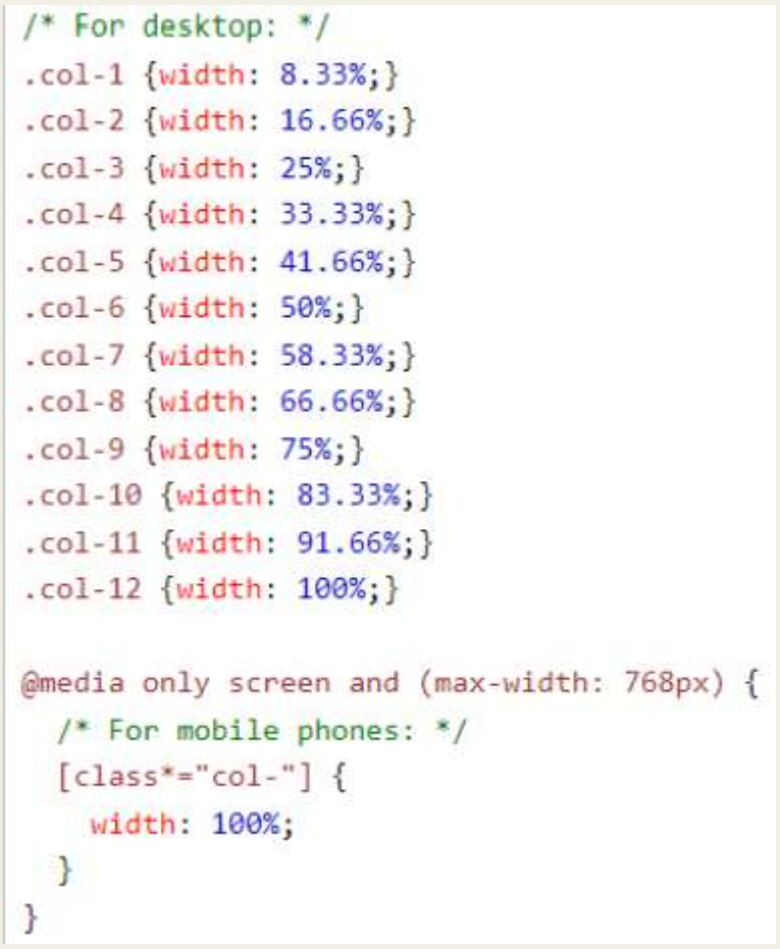


Figure 107 – *Media query (**Source**: https://www.w3schools.com/css/css_rwd_mediaqueries.asp)*

Use a media query to add a breakpoint at 768px:

Example: When the screen (browser window) gets smaller than 768px, each column should have a width of 100%:

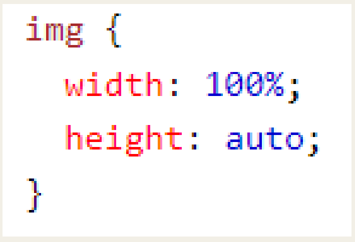


```
/* For desktop: */
.col-1 {width: 8.33%;}
.col-2 {width: 16.66%;}
.col-3 {width: 25%;}
.col-4 {width: 33.33%;}
.col-5 {width: 41.66%;}
.col-6 {width: 50%;}
.col-7 {width: 58.33%;}
.col-8 {width: 66.66%;}
.col-9 {width: 75%;}
.col-10 {width: 83.33%;}
.col-11 {width: 91.66%;}
.col-12 {width: 100%;}

@media only screen and (max-width: 768px) {
  /* For mobile phones: */
  [class*="col-"] {
    width: 100%;
  }
}
```

Figure 108 *– Media query (**Source**: https://www.w3schools.com/css/css_rwd_mediaqueries.asp)*

RWD Images

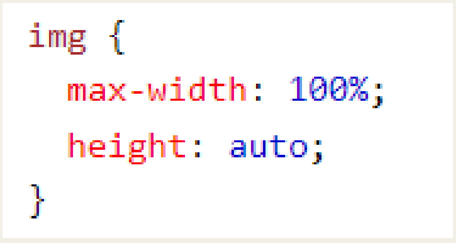
- If the width property is set to a percentage and the height property is set to "auto", the image will be responsive and scale up and down:



```
img {
  width: 100%;
  height: auto;
}
```

Figure 109 *– Image (**Source**: https://www.w3schools.com/css/css_rwd_images.asp)*

- If the max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:



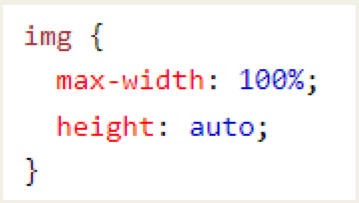
```
img {
  max-width: 100%;
  height: auto;
}
```

Figure 110 *– Image (**Source**: https://www.w3schools.com/css/css_rwd_images.asp)*

Notice that in the example above, the image can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

Using the max-width property:

The max-width property is set to 100%, the image will scale down if it has to, but never scale up to be larger than its original size:

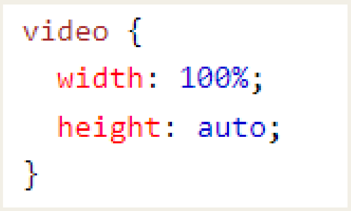


```css
img {
  max-width: 100%;
  height: auto;
}
```

***Figure 111** – Image (**Source**: https://www.w3schools.com/css/css_rwd_images.asp)*

RWD Videos

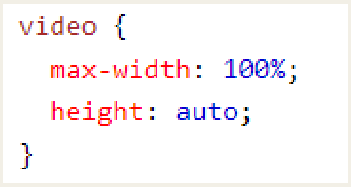
- If the width property is set to 100%, the video player will be responsive and scale up and down:



```css
video {
  width: 100%;
  height: auto;
}
```

***Figure 112** – Video (**Source**: https://www.w3schools.com/css/css_rwd_videos.asp)*

- If the max-width property is set to 100%, the video player will scale down if it must, but never scale up to be larger than its original size:



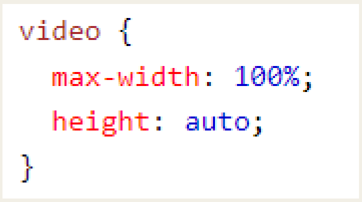
```
video {
  max-width: 100%;
  height: auto;
}
```

Figure 113 *– Video (**Source**: https://www.w3schools.com/css/css_rwd_videos.asp)*

Notice that in the example above, the video player can be scaled up to be larger than its original size. A better solution, in many cases, will be to use the max-width property instead.

Using the max-width property:

If the max-width property is set to 100%, the video player will scale down if it has to, but never scale up to be larger than its original size:



```
video {
  max-width: 100%;
  height: auto;
}
```

Figure 114 *– Video (**Source**: https://www.w3schools.com/css/css_rwd_videos.asp)*

RWD Frameworks

- There are many free CSS Frameworks that offer Responsive Design.
- A great way to create a responsive design, is to use a responsive style sheet, like W3.CSS
- W3.CSS makes it easy to develop sites that look nice at any size!
- Another popular framework is Bootstrap. It uses HTML and CSS to make responsive web pages.

CSS_Grid

Grid intro

- The CSS Grid Layout Module offers a grid-based layout system, with rows and columns, making it easier to design web pages without having to use floats and positioning.

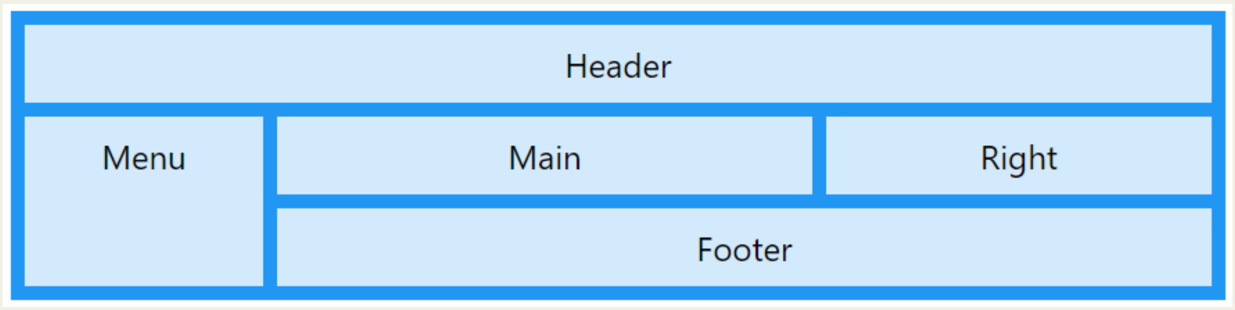


***Figure 115** – CSS Grid (**Source**: https://www.w3schools.com/css/css_grid.asp)*

Grid Container

- To make an HTML element behave as a grid container, you have to set the display property to grid or inline-grid
- Grid containers consist of grid items, placed inside columns and rows.

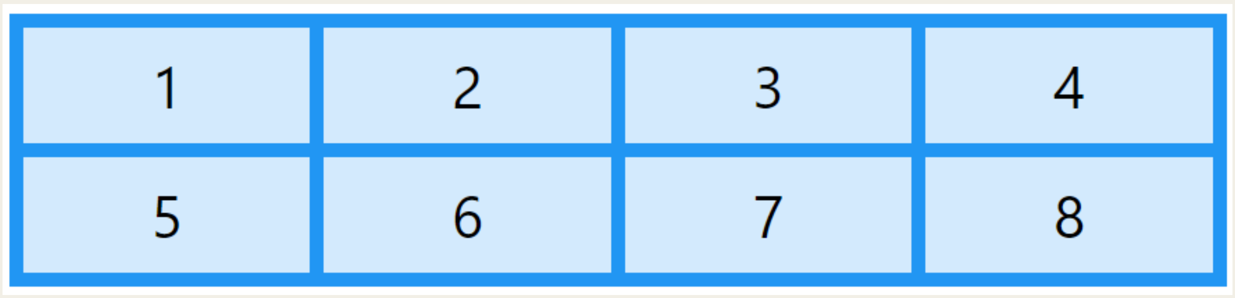
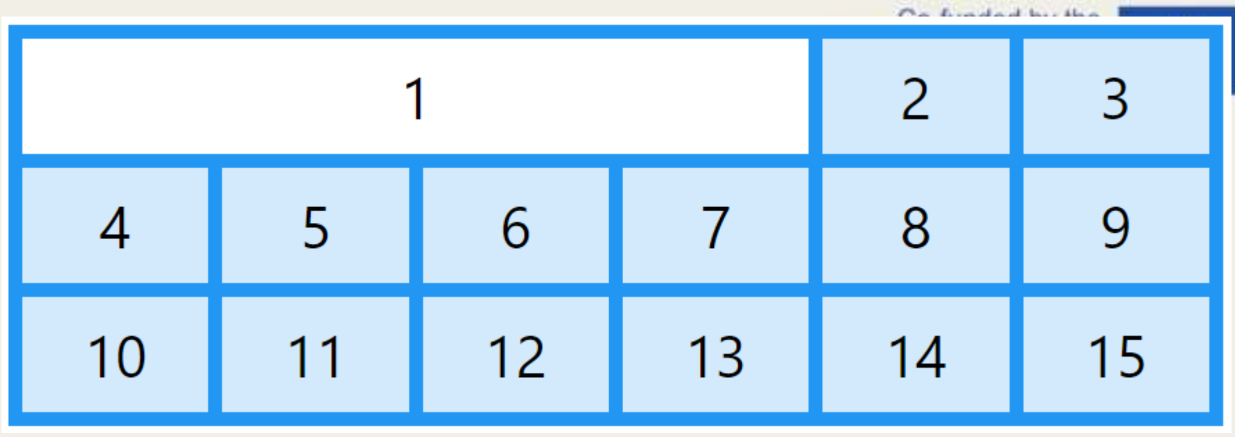


***Figure 116** – Grid Container (**Source**: https://www.w3schools.com/css/css_grid_container.asp)*

Grid Item

- A grid *container* contains grid *items*.
- By default, a container has one grid item for each column, in each row, but you can style the grid items so that they will span multiple columns and/or rows.
- The grid-column property defines on which column(s) to place an item.
- You define where the item will start, and where the item will end.



1				2	3
4	5	6	7	8	9
10	11	12	13	14	15

Figure 117 *– Grid Items (**Source**: https://www.w3schools.com/css/css_grid_item.asp)*

Co-funded by the Erasmus+ Programme of the European Union

Sass Introduction

What is Sass?

- Sass stands for Syntactically Awesome Stylesheet
- Sass is an extension to CSS
- Sass is a CSS pre-processor
- Sass is completely compatible with all versions of CSS
- Sass reduces repetition of CSS and therefore saves time
- Sass was designed by Hampton Catlin and developed by Natalie Weizenbaum in 2006
- Sass is free to download and use

Why Use Sass?

Stylesheets are getting larger, more complex, and harder to maintain. This is where a CSS pre-processor can help.

Sass lets you use features that do not exist in CSS, like variables, nested rules, mixins, imports, inheritance, built-in functions, and other stuff.

A Simple Example why Sass is Useful

Let's say we have a website with three main colors:

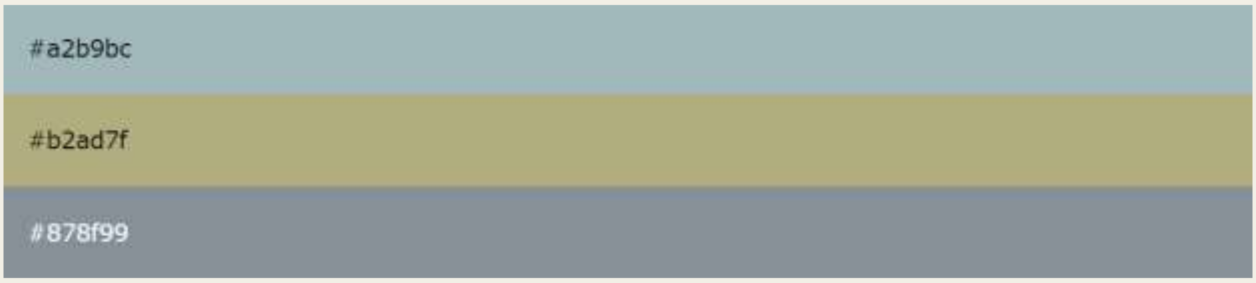
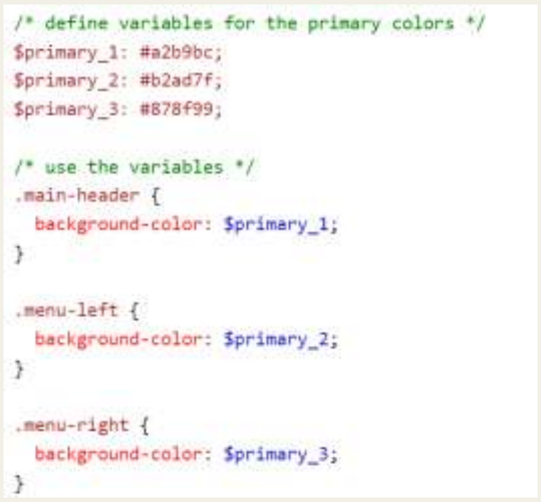


***Figure 118**– Colors (**Source**: https://www.w3schools.com/sass/sass_intro.php)*

So, how many times do you need to type those HEX values? A LOT of times. And what about variations of the same colors?

Instead of typing the above values a lot of times, you can use Sass and write this:



```
/* define variables for the primary colors */
$primary_1: #a2b9bc;
$primary_2: #b2ad7f;
$primary_3: #878f99;

/* use the variables */
.main-header {
  background-color: $primary_1;
}

.menu-left {
  background-color: $primary_2;
}

.menu-right {
  background-color: $primary_3;
}
```

Figure 119 *– SASS variable (**Source**: https://www.w3schools.com/sass/sass_variables.php)*

So, when using Sass, and the primary color changes, you only need to change it in one place.

How Does Sass Work?

A browser does not understand Sass code. Therefore, you will need a Sass pre-processor to convert Sass code into standard CSS.

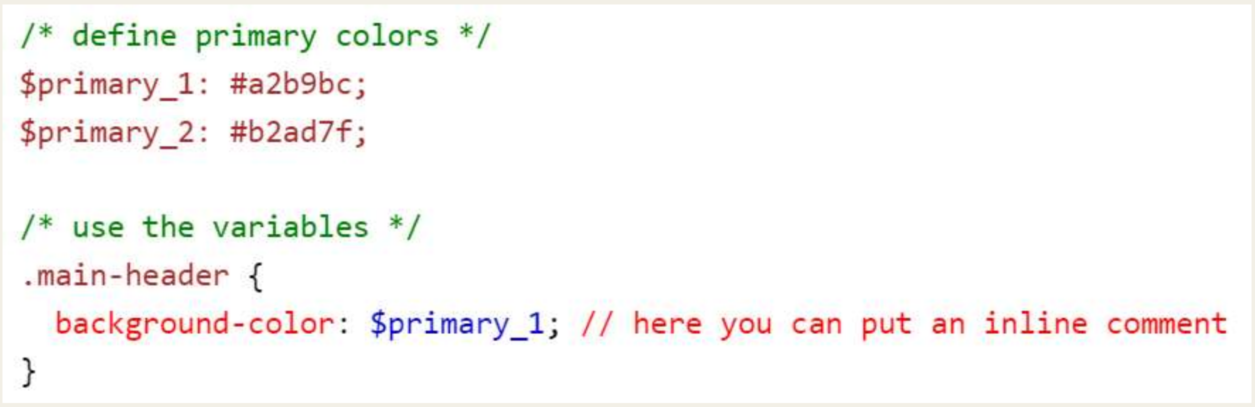
This process is called transpiling. So, you need to give a transpiler (some kind of program) some Sass code and then get some CSS code back.

Sass File Type

Sass files has the ".scss" file extension.

Sass Comments

Sass supports standard CSS comments /* comment */, and in addition it supports inline comments // comment:



```
/* define primary colors */
$primary_1: #a2b9bc;
$primary_2: #b2ad7f;

/* use the variables */
.main-header {
  background-color: $primary_1; // here you can put an inline comment
}
```

Figure 120 *– Comments (**Source**: https://www.w3schools.com/sass/sass_intro.php)*

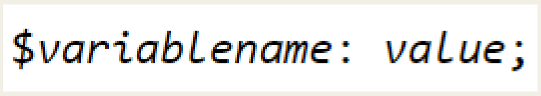
Sass Variables

Variables are a way to store information that you can re-use later.

With Sass, you can store information in variables, like:

- strings
- numbers
- colors
- booleans
- lists
- nulls

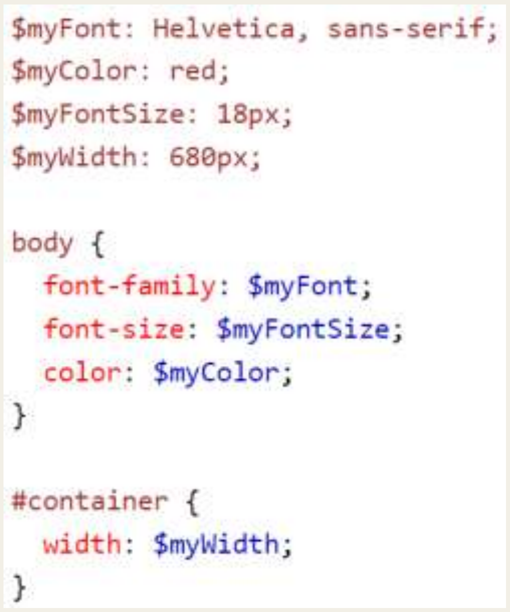
Sass uses the $ symbol, followed by a name, to declare variables:



```
$variablename: value;
```

Figure 121 *– SASS variable (**Source**: https://www.w3schools.com/sass/sass_variables.php)*

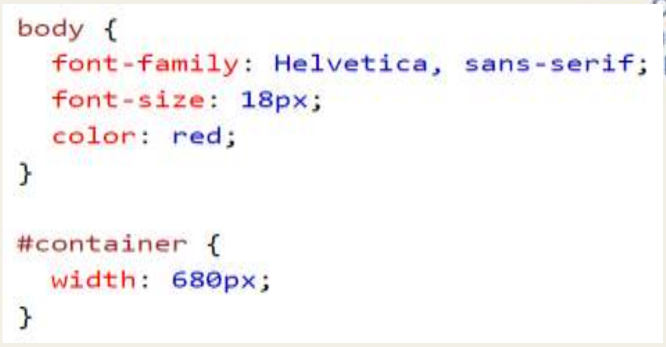
The following example declares 4 variables named myFont, myColor, myFontSize, and myWidth. After the variables are declared, you can use the variables wherever you want:



```
$myFont: Helvetica, sans-serif;
$myColor: red;
$myFontSize: 18px;
$myWidth: 680px;

body {
  font-family: $myFont;
  font-size: $myFontSize;
  color: $myColor;
}

#container {
  width: $myWidth;
}
```

Figure 122 *– SASS variable (**Source**: https://www.w3schools.com/sass/sass_variables.php)*

So, when the Sass file is transpiled, it takes the variables (myFont, myColor, etc.) and outputs normal CSS with the variable values placed in the CSS, like this:

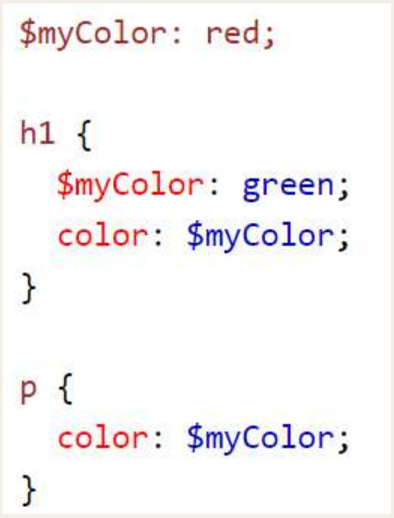


```
body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}

#container {
  width: 680px;
}
```

__Figure 123__ – SASS variable (__Source__: https://www.w3schools.com/sass/sass_variables.php)

Sass Variable Scope

Sass variables are only available at the level of nesting where they are defined. Look at the following example:



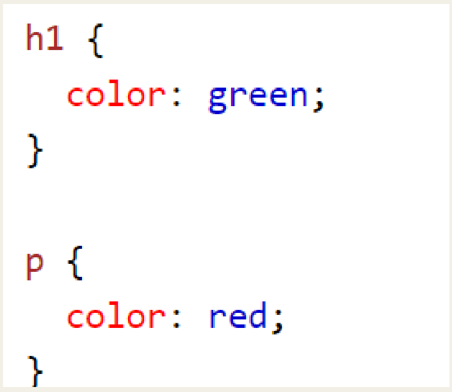
```
$myColor: red;

h1 {
  $myColor: green;
  color: $myColor;
}

p {
  color: $myColor;
}
```

__Figure 124__ – SASS variable (__Source__: https://www.w3schools.com/sass/sass_variables.php)

Will the color of the text inside a <p> tag be red or green? It will be red!

The other definition, $myColor: green; is inside the <h1> rule, and will only be available there!

So, the CSS output will be:



```
h1 {
  color: green;
}

p {
  color: red;
}
```

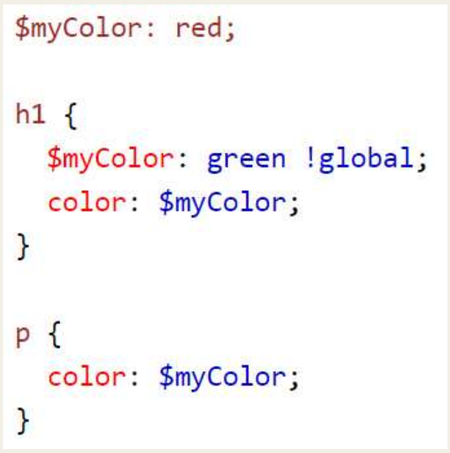
Figure 125 *– SASS variable (**Source**: https://www.w3schools.com/sass/sass_variables.php)*

Ok, that is the default behavior for variable scope.

Using Sass !global

The default behavior for variable scope can be overridden by using the !global switch. !global indicates that a variable is global, which means that it is accessible on all levels.

Look at the following example (same as above; but with !global added):

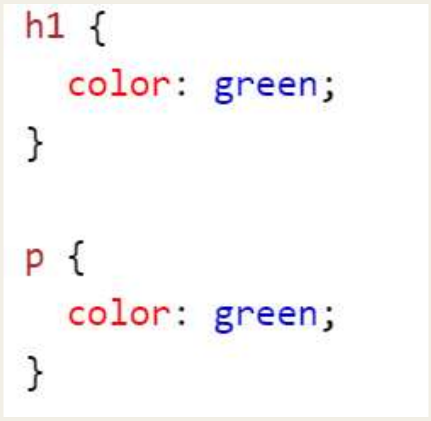


```
$myColor: red;

h1 {
  $myColor: green !global;
  color: $myColor;
}

p {
  color: $myColor;
}
```

Figure 126 *– SASS variable (**Source**: https://www.w3schools.com/sass/sass_variables.php)*

Now the color of the text inside a <p> tag will be green!

So, the CSS output will be:



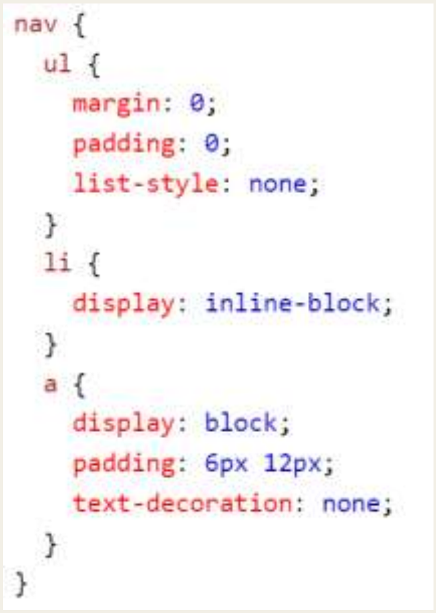
```
h1 {
  color: green;
}

p {
  color: green;
}
```

***Figure 127** – SASS variable (**Source**: https://www.w3schools.com/sass/sass_variables.php)*

Sass Nested Rules

Sass lets you nest CSS selectors in the same way as HTML.

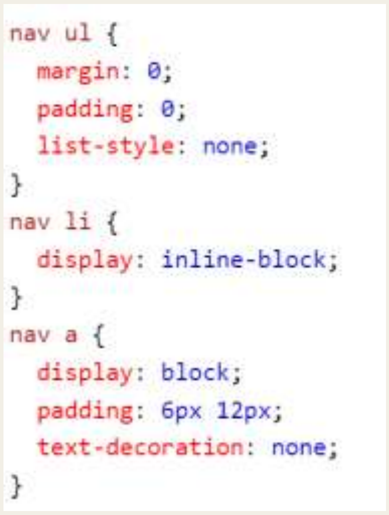
Look at an example of some Sass code for a site's navigation:



```
nav {
  ul {
    margin: 0;
    padding: 0;
    list-style: none;
  }
  li {
    display: inline-block;
  }
  a {
    display: block;
    padding: 6px 12px;
    text-decoration: none;
  }
}
```

***Figure 128** – SASS nested (**Source**: https://www.w3schools.com/sass/sass_nesting.php)*

Notice that in Sass, the ul, li, and selectors are nested inside the nav selector.
While in CSS, the rules are defined one by one (not nested):



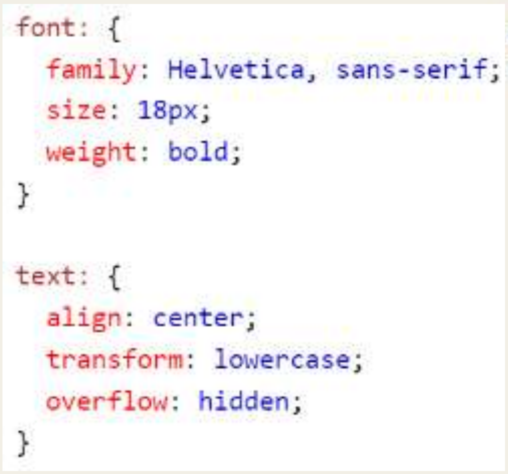
```
nav ul {
  margin: 0;
  padding: 0;
  list-style: none;
}
nav li {
  display: inline-block;
}
nav a {
  display: block;
  padding: 6px 12px;
  text-decoration: none;
}
```

Figure 129 *– SASS nested (**Source**: https://www.w3schools.com/sass/sass_nesting.php)*

Because you can nest properties in Sass, it is cleaner and easier to read than standard CSS.

Sass Nested Properties

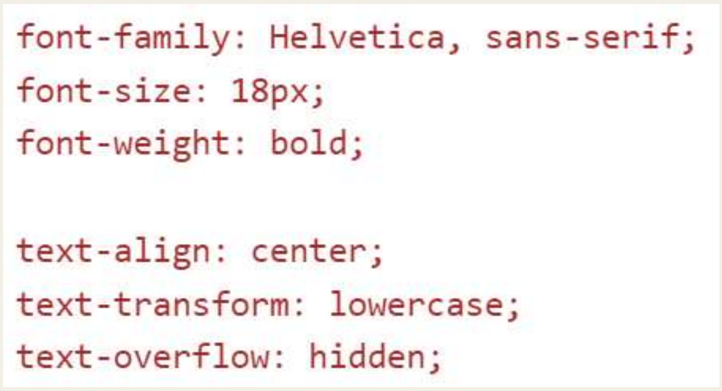
Many CSS properties have the same prefix, like font-family, font-size and font-weight or text-align, text-transform and text-overflow.
With Sass, you can write them as nested properties:



```
font: {
  family: Helvetica, sans-serif;
  size: 18px;
  weight: bold;
}

text: {
  align: center;
  transform: lowercase;
  overflow: hidden;
}
```

***Figure 130** – SASS nested (**Source**: https://www.w3schools.com/sass/sass_nesting.php)*

The Sass transpiler will convert the above to normal CSS:



```
font-family: Helvetica, sans-serif;
font-size: 18px;
font-weight: bold;

text-align: center;
text-transform: lowercase;
text-overflow: hidden;
```

***Figure 131** – SASS nested (**Source**: https://www.w3schools.com/sass/sass_nesting.php)*

Sass @import and Partials

Sass keeps the CSS code DRY (Don't Repeat Yourself). One way to write DRY code is to keep related code in separate files.

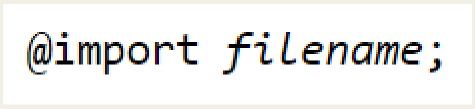
You can create small files with CSS snippets to include in other Sass files. Examples of such files can be: reset file, variables, colors, fonts, font-sizes, etc.

Sass Importing Files

Just like CSS, Sass also supports the @import directive.

The @import directive allows you to include the content of one file in another.

The CSS @import directive has a major drawback due to performance issues; it creates an extra HTTP request each time you call it. However, the Sass @import directive includes the file in the CSS; so no extra HTTP call is required at runtime!

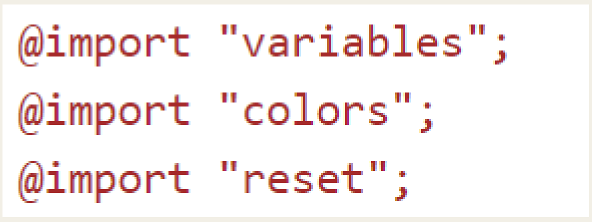


```
@import filename;
```

Figure 132 *– SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

Tip: You do not need to specify a file extension, Sass automatically assumes that you mean a .sass or .scss file. You can also import CSS files. The @import directive imports the file and any variables or mixins defined in the imported file can then be used in the main file.

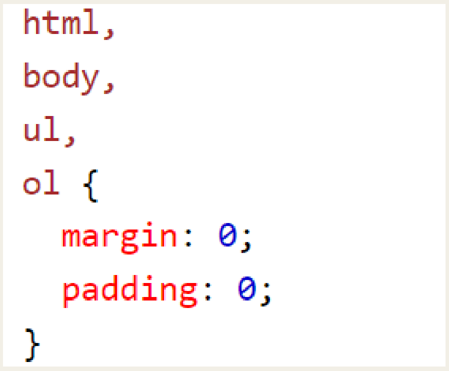
You can import as many files as you need in the main file:



```
@import "variables";
@import "colors";
@import "reset";
```

Figure 133 *– SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

Let's look at an example: Let's assume we have a reset file called "reset.scss", that looks like this:

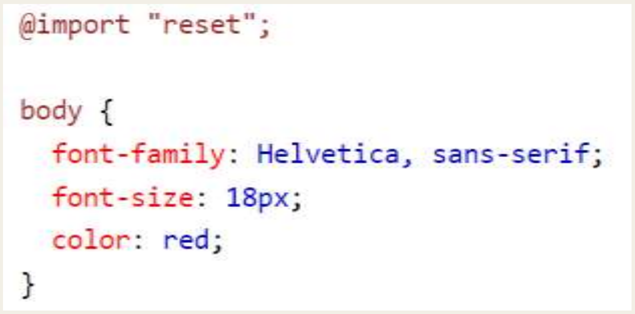


```
html,
body,
ul,
ol {
  margin: 0;
  padding: 0;
}
```

Figure 134 *– SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

And now we want to import the "reset.scss" file into another file called "standard.scss".

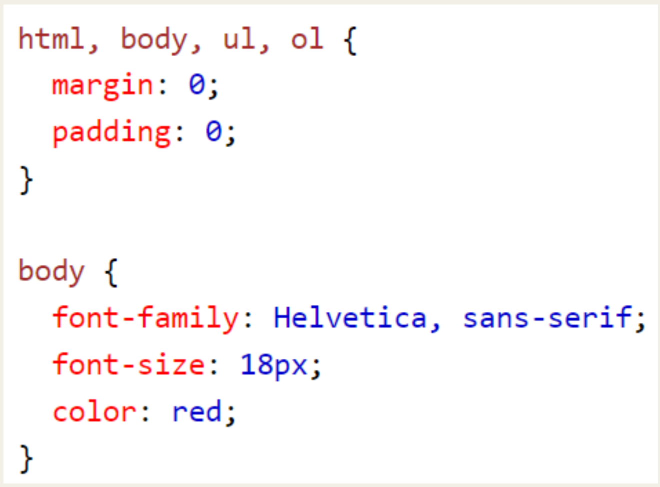
Here is how we do it: It is normal to add the @import directive at the top of a file; this way its content will have a global scope:



```
@import "reset";

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}
```

Figure 135 *– SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

So, when the "standard.css" file is transpiled, the CSS will look like this:



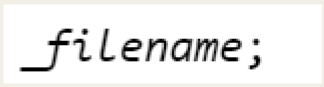
```css
html, body, ul, ol {
  margin: 0;
  padding: 0;
}

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: red;
}
```

Figure 136 *– SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

Sass Partials

By default, Sass transpiles all the .scss files directly. However, when you want to import a file, you do not need the file to be transpiled directly.
Sass has a mechanism for this: If you start the filename with an underscore, Sass will not transpile it. Files named this way are called partials in Sass.

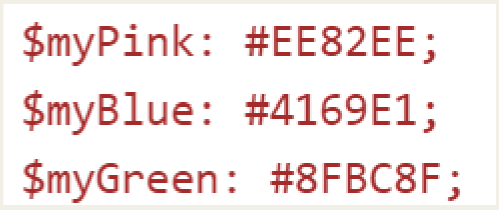
So, a partial Sass file is named with a leading underscore:



```
_filename;
```

Figure 137 *– SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

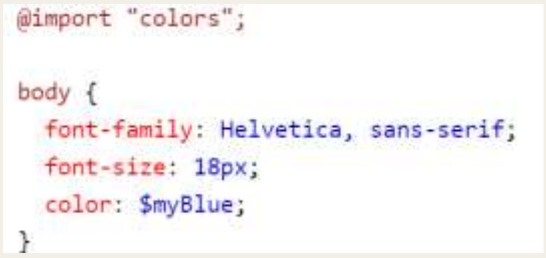
The following example shows a partial Sass file named "_colors.scss". (This file will not be transpiled directly to "colors.css"):



```
$myPink: #EE82EE;
$myBlue: #4169E1;
$myGreen: #8FBC8F;
```

***Figure 138** – SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

Now, if you import the partial file, omit the underscore. Sass understands that it should import the file "_colors.scss":



```
@import "colors";

body {
  font-family: Helvetica, sans-serif;
  font-size: 18px;
  color: $myBlue;
}
```

***Figure 139** – SASS import (**Source**: https://www.w3schools.com/sass/sass_import.php)*

The European Commission's support for the production of this publication does not constitute an endorsement of the contents, which reflect the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Sass @mixin and @include

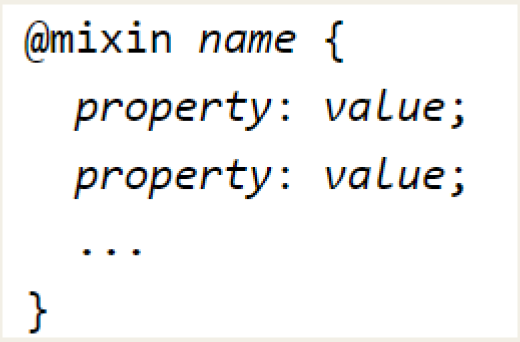
Sass Mixins

The @mixin directive lets you create CSS code that is to be reused throughout the website.

The @include directive is created to let you use (include) the mixin.

Defining a Mixin

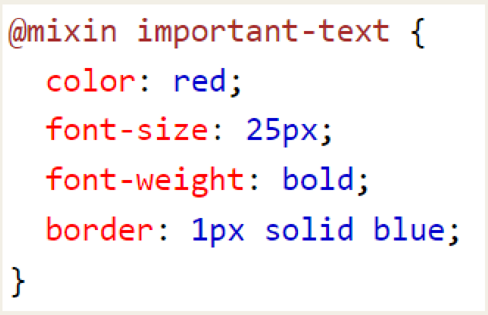
A mixin is defined with the @mixin directive.



```
@mixin name {
  property: value;
  property: value;
  ...
}
```

***Figure 140** – sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

The following example creates a mixin named "important-text":

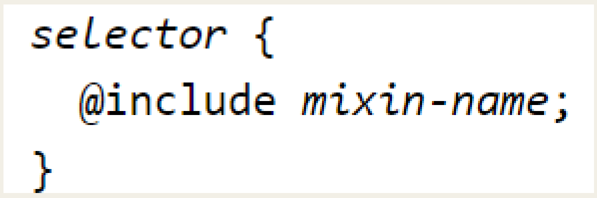


```
@mixin important-text {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
}
```

***Figure 141** – sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

Using a Mixin

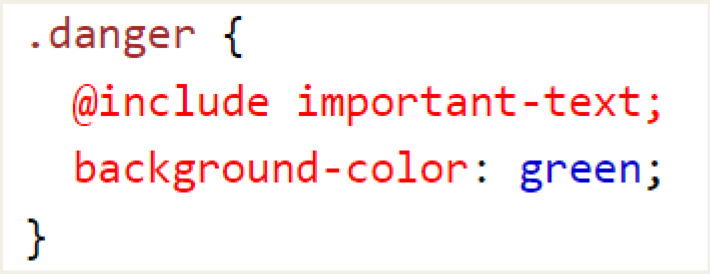
The @include directive is used to include a mixin.



```
selector {
  @include mixin-name;
}
```

Figure 142 *– sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

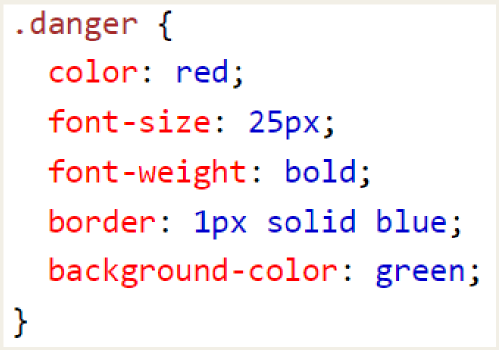
So, to include the important-text mixin created above:



```
.danger {
  @include important-text;
  background-color: green;
}
```

Figure 143 *– sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

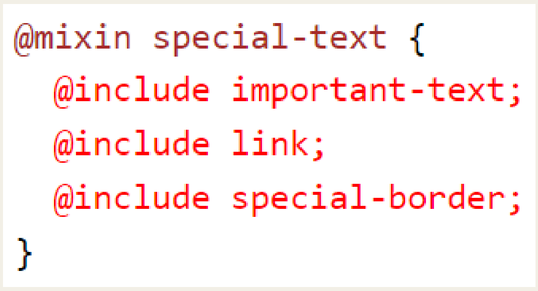
The Sass transpiler will convert the above to normal CSS:



```
.danger {
  color: red;
  font-size: 25px;
  font-weight: bold;
  border: 1px solid blue;
  background-color: green;
}
```

Figure 144 *– sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

A mixin can also include other mixins:



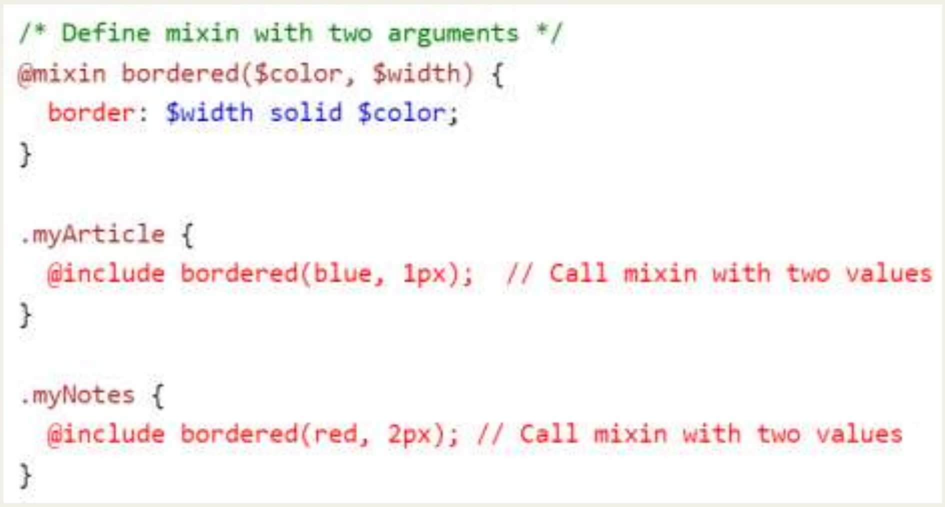
```
@mixin special-text {
  @include important-text;
  @include link;
  @include special-border;
}
```

***Figure 145** – sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

Passing Variables to a Mixin

Mixins accept arguments. This way you can pass variables to a mixin.

Here is how to define a mixin with arguments:

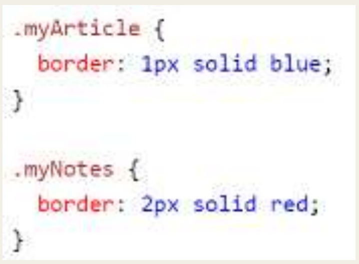


```
/* Define mixin with two arguments */
@mixin bordered($color, $width) {
  border: $width solid $color;
}

.myArticle {
  @include bordered(blue, 1px);  // Call mixin with two values
}

.myNotes {
  @include bordered(red, 2px); // Call mixin with two values
}
```

***Figure 146** – sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

Notice that the arguments are set as variables and then used as the values (color and width) of the border property.

After compilation, the CSS will look like this:



```
.myArticle {
  border: 1px solid blue;
}

.myNotes {
  border: 2px solid red;
}
```

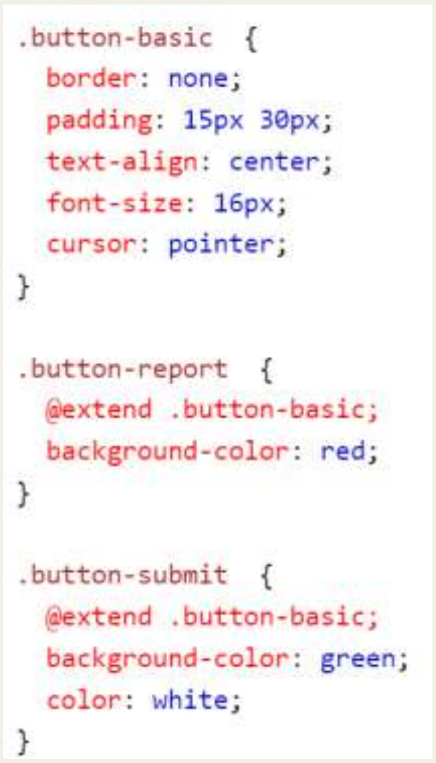
Figure 147 *– sass @mixin (**Source**: https://www.w3schools.com/sass/sass_mixin_include.php)*

Sass @extend Directive

The @extend directive lets you share a set of CSS properties from one selector to another.

The @extend directive is useful if you have almost identically styled elements that only differ in some small details.

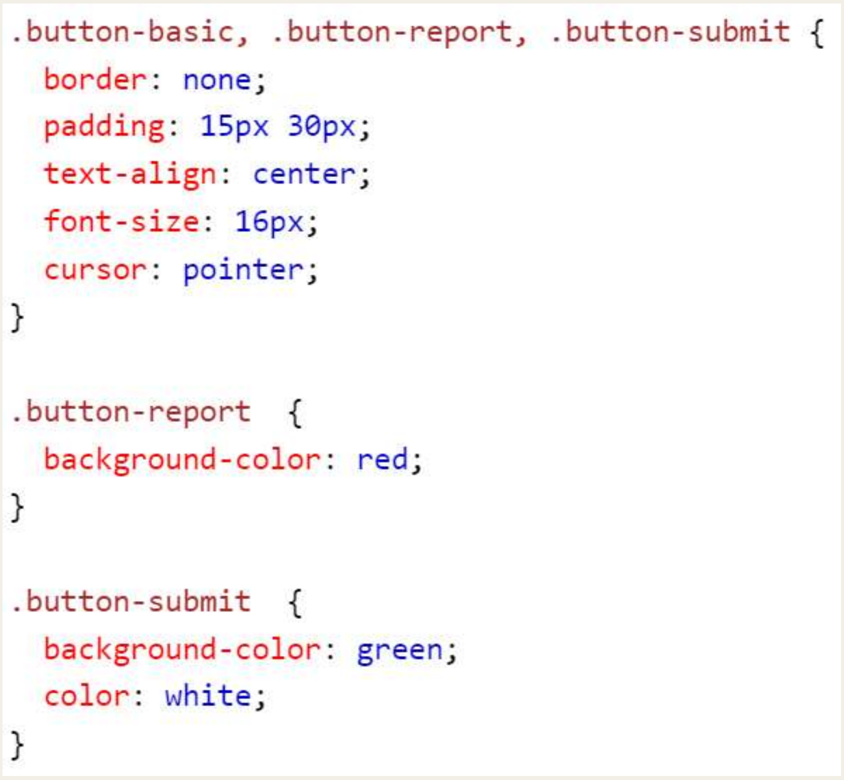
The following Sass example first creates a basic style for buttons (this style will be used for most buttons). Then, we create one style for a "Report" button and one style for a "Submit" button. Both "Report" and "Submit" button inherit all the CSS properties from the .button-basic class, through the @extend directive. In addition, they have their own colors defined:



```
.button-basic  {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report  {
  @extend .button-basic;
  background-color: red;
}

.button-submit  {
  @extend .button-basic;
  background-color: green;
  color: white;
}
```

***Figure 148** – SASS Extend (**Source**: https://www.w3schools.com/sass/sass_extend.php)*

After compilation, the CSS will look like this:



```
.button-basic, .button-report, .button-submit {
  border: none;
  padding: 15px 30px;
  text-align: center;
  font-size: 16px;
  cursor: pointer;
}

.button-report  {
  background-color: red;
}

.button-submit  {
  background-color: green;
  color: white;
}
```

Figure 149 *– SASS Extend (**Source**: https://www.w3schools.com/sass/sass_extend.php)*

By using the @extend directive, you do not need to specify several classes for an element in your HTML code, like this: <button class="button-basic button-

report">Report this</button>. You just need to specify .button-report to get both sets of styles.

The @extend directive helps keep your Sass code very DRY.