CSS

 Cascading Style Sheets, is a language that web developers use to style the HTML content on a web page.

To style an HTML element, you can add the style attribute directly to the **opening tag**. After you add the attribute, you can set it equal to the CSS style(s) you'd like applied to that element.

HTML allows you to write CSS code in its own dedicated section with the **<style>** element. CSS can be written between opening and closing <style> tags and it must be placed inside of the <head> element.

With a CSS file, you can write all the CSS code needed to style a page without sacrificing the readability and maintainability of your HTML file. You can use the **<link>** element to link HTML and CSS files together and must be placed within the head of the HTML file. It also requires the attributes href, type and rel.

 A tag name is the word (or character) between HTML angle brackets. CSS can select HTML elements by using an **element's tag name**.

HTML can have attributes.  It's also possible to select an element by its **class** attribute.

If an HTML element needs to be styled uniquely, we can add an **ID** to the element placing an ID attribute. While classes are meant to be used many times, an ID is meant to style only one element

Specificity is the order by which the browser decides which CSS styles will be displayed. IDs are the most specific selector in CSS, followed by classes, and finally, tags.

Chaining is combining multiple selectors at the same time. It’s done adding **.style** for an element.

CSS also supports selecting elements that are **nested** within other HTML elements.

HTML:

<p style= “property: value; property: value”>Text</p> #directly in the opening tag

<head> #inside head element

<style>

p{

property: value;

}

</style>

</head>

<head>

<link

href=“link” #addres to the CSS file

type=“text/css” #describes the type of document linking to

rel=“stylesheet” #describes the relation between the HTML

> file and CSS

</head>

<element class=“example, example”>Text</element> #selects a class

<element id=“example”></element>

CSS:

Tag\_element {property: value; }

.classname{} #creates a class

#id-name{}

element.special{} #selects only the elements that have a class of

special

.classname nested\_tag\_element{} #applies the class to the nested element

Tag\_element, .classname{} #adds the style to multiple selectors

-Visual rules:

Font refers to the technical term [typeface](https://en.wikipedia.org/wiki/Typeface), or font family. The font specified in a stylesheet must be installed on a user's computer for that font to display when a user visits the web page. The default typeface for all HTML elements is Times New Roman. It's a good practice to limit the number of typefaces used on a web page to 2 or 3. This helps the page load faster in some cases and is usually a good design decision. When the name of a typeface consists of more than one word, it's a best practice to enclose the typeface's name in quotes.

The size of the text is measure in pixels (px).

Opacity can be used to make elements fade into others for a nice overlay effect. It's measured from 0 to 1, with 1 representing 100%, or fully visible and opaque, and 0 representing 0%, or fully invisible.

CSS

tag{

font-family: “example”; #change the typeface of text

font-size: numberpx; #change the size of text

font-weight: **bold**/normal; #controls how bold or thin text appears

text-align: right/center/left;

color: example;

background-color: example;

opacity: number;

background-image: url(“link”); #set the element background to display an

image

background-image: url(“folder/image.jpg”); #link to an image inside an existing project

}

-Box Model:

All elements on a web page are interpreted by the browser as "living" inside of a box. This is what is meant by the **box model**. It comprises the set of properties which define parts of an element that take up space on a web page.

An element's content has two dimensions: a **height** and a **width**. By default, the dimensions of an HTML box are set to hold the raw contents of the box. They can be set in pixels or percentage.

A **border** is a line that surrounds an element, like a frame around a painting. Borders can be set with a specific width, style and color. Style is the design of the border, web browsers can render any of 10 different styles including none, dotted, solid, etc. The default border is medium none color.

You can create a border that is a perfect circle by setting the **radius** equal to the height of the box, or to 100%.

**Padding** is the space between the contents of a box and the borders of a box, like the space between a picture and the frame surrounding it. Each property affects the padding on only one side of the box’s content.

**Margin** refers to the space directly outside of the box.  It's common to see margin values used for a specific side of an element.  The margin property also lets you **center content**. The auto value instructs the browser to adjust the left and right margins until the element is centered within its containing element. In order to center an element, a width must be set for that element. Otherwise, the width of the element will be automatically set to the full width of its containing element. It's not possible to center an element that takes up the full width of the page.

One additional difference between padding and margin is that top and bottom margins, also called vertical margins, **collapse**, while top and bottom padding does not. Horizontal margins (left and right), like padding, are always displayed and added together. For example, if two elements are next to each other, they will be as far apart as the **sum** of their adjacent margins. Unlike horizontal margins, vertical margins do not add. Instead, the larger of the two vertical margins sets the distance between adjacent elements.

Because a web page can be viewed through displays of differing screen size, the content on the web page can suffer from those changes in size. To avoid this problem, CSS offers two properties that can limit how narrow or how wide an element's box can be sized to: **min-width** and **max-width**. You can also limit the **minimum** and **maximum height** of an element.

The **overflow** property controls what happens to content that spills, or overflows, outside its box. It can be set to one of the following values: **hidden**, any content that overflows will be hidden from view; **scroll**: a scrollbar will be added to the element’s box; and **visible**: the default value, the overflow value will be displayed outside of the containing element. The overflow property is set on a parent element to instruct a web browser how to render child elements.

In the default box model, content-box, box dimensions are affected by border thickness and padding (height or width = height/width + padding + border) . To change this, box-sizing property controls the box model used by the browser. In the border-box model, the border thickness and padding would remain entirely inside of the box.

CSS

tag{

height: npx;

width: npx;

border: npx/thin/medium/thick style-example color-example; #width, style and color

border-radius: npx #set all 4 corners to the radius

padding: npx;

padding-top/right/bottom/left: npx;

padding: topnumber rightnumber bottonnumber leftnumber; #all four sides must be specified

padding: topandbottomnumber rightandleftnumber; #2 numbers must be provide

margin: npx;

margin-top/right/bottom/left: npx;

margin: topnumber rightnumber bottonnumber leftnumber; #all four sides must be specified

margin: topandbottomnumber rightandleftnumber; #2 numbers must be provide

width: 400px; #centers the element

margin: 0 auto;

min-width: npx;

max-width: npx;

min-height: npx;

max-height: npx;

overflow: hidden/scroll/visible;

visibility: hidden/visible; #hides only the content of the element,

leaving an empty space

}

\*{

box-sizing: content-box/border-box;

}

-Display and positioning:

CSS includes properties that change how a browser positions elements. These properties specify where an element is located on a page, if the element can share lines with other elements, and other related attributes.

The default position of an element can be changed by setting its **position** property. **Static** is the default property and it doesn’t need to be set.

The **relative** property allows you to position an element relative to its default static position on the web page. But to specify where the div should be positioned on the page the four **offset properties** are needed.

When set to **absolute**, an element's position is relative to its closest positioned parent element. All other elements on the page will ignore the element and act like it is not present on the page. It can be pinned to any part of the web page, but the element will still move with the rest of the document when the page is scrolled.

An element can be fixed to a specific position on the page, regardless of user scrolling, by setting its position to **fixed**.

The **z-index** of an element specifies how far back or how far forward an element appears on the page when it overlaps other elements. Depending on their values, the integers instruct the browser on the order in which elements should be displayed on the web page.

Every HTML element has a default display value that dictates if it can share horizontal space with other elements. The **display** property allows you control how an element flows vertically and horizontally a document.

Inline elements have a box that wraps tightly around their content, only taking up the amount of space necessary to display their content and not requiring a new line after each element. They cannot have manually-adjusted width or height. The CSS display property provides the ability to make any element an **inline** element. This includes elements that are not inline. **Block** elements take up the width of their container. These elements fill the entire width of the page by default, but their width property can also be set. Unless otherwise specified, they are the height necessary to accommodate their content. **Inline-block** display combines features of both inline and block elements, images for example. Inline-block elements can have set width and height, but they can also appear next to each other and do not take up their entire container width.

The **float** property can move elements as far left or as far right as possible on a web page.

The **clear** property specifies how elements should behave when they bump into each other on the page. It can take on one of the specified values clearing that side.

CSS

tag{

position: static/relative/absolute/fixed;

top/bottom/left/right: npx; #offset properties

z-index: integer;

display: inline/block/inline-block;

float: left/right;

clear: left/right/both/none;

}

-Color:

There are four ways to represent color in CSS.

**Named** colors, there are 147.

**Hexadecimal** or hex colors, begins with a hash character (#) which is followed by three or six characters. The characters represent values for red, blue and green. The hexadecimal number system has 16 digits (0-15) represented by numbers and letters.

**Rgb** colors use decimal numbers from 0 to 255. Each of the three values represents a color component: red, green and blue in that order.

The **HSL** or the hue-saturation-lightness color scheme uses numbers that represents a hue. The first number refers to an angle on a color wheel and can be between 0 and 360. The second refers to the intensity or purity of the color. And the third refers to how light or dark the color is.

It is possible to add **opacity** to color in RGB and HSL by adding a fourth value, **alpha** (a), which is represented by a decimal number from zero to one.

CSS

tag{

color/background-color:

name; #named color

#8FBC5F; #hex color

rgb/rgba(number, number, number /, number ); #rgb color / rgba

hsl/hsla(number, n%, n% /, number); #hue, saturation, lightness

}

-Typography:

Typography is the art of arranging text on a page.

The **font-weight** property can also be assigned a number value to style text on a numeric scale ranging from 100 to 900. Valid values are multiples of 100 within this range. 400 is the default for most texts, 700 signifies bold and 300, light. Not all fonts can be assigned a numeric font-weight. You can look up the font you are using to see which font-weight values are available.

The **line-height** property is use to set how tall we want the line containing our text to be, regardless of the height of the text. If it takes a unitless number will be an absolute value that will compute the line height as a ratio of the font size. The unitless ratio value is the preferred method, since it is responsive and based exclusively on the current font size.

**Serif** fonts have extra details on the ends of each letter. **Sans-Serif** fonts do not.

Most computers have a small set of typefaces pre-installed., including serif and sans-serif fonts. These pre-installed fonts serve as **fallback fonts** if the stylesheet specifies a font which is not installed on a user's computer.

New fonts are often centralized in directories made available for public use. We refer to these fonts as non-user fonts

CSS offers a way to import fonts directly into stylesheets with the **@font-face** property. It is important to stress the need to copy the @font-face rules to the top of the stylesheet for the font to load correctly in the project.

Local fonts can be added to a document with the @font-face property and the path to the font's source.

HTML

<head>

<link href=“new font link” type=“tet/css” rel=“stylesheet”>

</head>

CSS

tag{

font-weight: bold/normal/number;

word-spacing: numberem; #increase the space between words

letter-spacing: numberem; #increase the space between letters

line-height: number;

font-family: example, backupfontexample, serif/sans-serif;

}

-Grids:

To set up a grid, you need to have both a grid container and grid items. The grid container will be a parent element that contains grid items as children and applies overarching styling and positioning to them.

To turn an HTML element into a grid container, you must set the element's **display** property to **grid**.

The **repeat** (repeat (number-of-rows/columns, size)) function will duplicate the specifications for rows or columns a given number of times. The second parameter can have multiple values.

It’s ´possible to make single grid items take up multiple rows and columns with the **grid-row**, **grid column** and **grid-area** properties. The keyword **span** is used to start or end a column or row relative to its other end.

HTML

<element class=“grid”>

<element class= “exampleclass”></element>

</element>

CSS:

.grid{

display: grid;

grid-template: rowsize rowsize / columnsize columnsize; #defines the numbers and size of

rows/column

grid-gap: rawsize columnsize; #puts blankspace between row/column

}

.exampleclass{

grid-row: grid-row-start / grid-row-end;

grid-column: grid-column-start/grid-column-end;

grid-area: grid-row-start / grid-column-start / grid-row-end / grid-column-end;

}