CSS REFERENCE

CSS describes how HTML elements are to be displayed on screen. External stylesheets are stored in CSS files, allowing a single file to control the presentation of multiple HTML pages.

A CSS rule consists of a selector, which refers to the HTML element or elements to which the rule applies, and a declaration block:

```
p {color: red;}
```

The declaration block contains one or more declarations separated by semicolons. Each declaration includes a CSS property name and a value, separated by a colon. Multiple CSS declarations are separated with semicolons, and declaration blocks are surrounded by curly braces.

CSS SELECTORS

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

TYPE OF SELECTOR	CSS RULE EXAMPLE	WHAT IT SELECTS
Element selector	`p { color: red; }`	*All* `<p>` *elements in HTML*
Class selector	`p.example { color: red; }`	*All* `<p class="example">` *elements in HTML*
ID selector	`p#example { color: red; }`	*The* `<p id="example">` *elements in HTML*
Multiple selector	`p,figcaption,a { color: red; }`	*All* `<p>`, `<figcaption>` *and* `<a>` *elements in HTML*
Descendant selector	`p.example span { color: red; }`	`<span>` *elements that are children of* `<p class="example">`
Pseudoclass selector	`a:hover { color: red; }`	*All* `<a>` *elements in HTML when the user hovers over them*

HOW TO ADD CSS

There are three ways to add CSS rules to HTML documents.

TYPE OF CSS	HOW IT IS ADDED	EXAMPLE
External stylesheet	Via a `<link>` in the HTML's `<head>` element	`<link href="style.css" rel="stylesheet">`
Internal stylesheet	Via a `<style>` tag in the HTML's `<head>` element	`<style> p {color:red;} </style>`
Inline style	Via a style attribute in the HTML tag	`<p style="color:red;">`

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

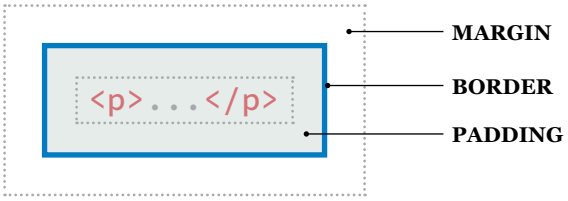
Consider using an **internal** stylesheet if one single HTML page has a unique style.

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

Order matters! If using multiple external stylesheets, the last declared style for an element will be rendered.

CSS BOX MODEL

All HTML elements can be considered as boxes. The CSS box model is essentially a box that wraps around every HTML element, and relates to design and layout. It consists of margins, borders, padding and the actual content. Padding creates extra space within an element, while margin creates extra space around an element.



EXAMPLE	VALUES	WHAT IT DOES
`margin-bottom: 30px;`	length, %, auto	*Adds space below an element before another appears*
`padding-top: 30px;`	length, %	*Adds space around an element to the edge of its box*
`border-width: 2px;`	thin, medium, thick, length	*Sets the width of the border on an element*
`border-color: red;`	color name, RGB, hexadecimal	*Sets the color of the border*
`border-style: solid;`	dotted, dashed, solid, double, groove, ridge	*Sets the color of the border*
`border-radius: 50%;`	length, %	*Rounds the corners of the element*

SHORTHANDS

You can use margin and padding as shorthand with one, two, three or four values. One value means all four sizes are the same. Two means the first is top and bottom, the second left and right. With three values, the order is top, left and right, then bottom. And four values means the values are in this order: top, right, bottom, left.

```
padding: 50px;   padding: 50px 20px;   padding: 50px 20px 10px;   padding: 50px 20px 10px 60px;
```

The border shorthand is used for border-width, border-style and border-color, in that order.

```
border: 5px solid red;                    outline: 5px solid red;
```

The outline shorthand is similar. But outlines and borders are not the same! Unlike a border, the outline is drawn *outside* the element's border, and may overlap other content. Also, the outline is NOT a part of the element's dimensions; the element's total width and height is not affected by the width of the outline.

CSS UNITS

CSS has several units for expressing length. Many properties take "length" values: width, margin, padding, font-size, etc. Always express these values without spaces between the number and unit.

UNITS	EXAMPLE	WHAT IT MEANS
px	`margin: 30px;`	*Sets size in number of pixels*
em	`font-size: 2em;`	*Relative to the font-size of the element (2em means 2 times the size of the current font)*
rem	`font-size: 2rem;`	*Relative to font-size of the root element*
vw	`width: 50vw;`	*Relative to 1% of the width of the viewport (i.e. the browser window size; 50=50%)*
%	`width: 50%;`	*Relative to the size (width in this case) of the parent element*

CSS COLORS

Colors in CSS can be specified by the following methods:

VALUES	EXAMPLE	HOW IT WORKS
Color name	`background-color: darkolive;`	*Uses one of 140 saved color names*
Hexadecimal value	`background-color: #31a62b;`	*Uses color's hexadecimal value (base 16) with hashtag*
RGB values	`background-color: rgb(114,12,200);`	*Uses color's RGB value in this order: red, green, blue*
RGBA values	`background-color: rgb(114,12,200,0.5);`	*Same as RGB, adding alpha (0-1) as fourth value*

For a complete list of CSS color names, go to **https://www.w3schools.com/cssref/css_colors.asp**

CSS BACKGROUNDS

The CSS background properties are used to define the background effects for elements.

EXAMPLE	VALUES	WHAT IT DOES
`background-color: #31a62b;`	colorname, rgb or hex	*Sets background color of element*
`background-image: url(../images/example.jpg);`	url()	*Sets background image of element*
`background-position: center top;`	left, right, center, top, bottom*	*Positions a background image*
`background-repeat: no-repeat;`	repeat-x, repeat-y, no-repeat	*Whether background image repeats*
`background-attachment: fixed;`	fixed, scroll	*Whether background image scrolls*
`background-size: cover;`	cover, length, %	*Sets size of background-image*

Values are in pairs, horizontal position (left, right or center) then the vertical (top, bottom or center). If only one value is used, the other will be "center."

SHORTHAND

You can set several background properties in one declaration, in the following order: background-color, background-image, background-repeat, background-position, background-attachment.

```
background: #31a62b url(../images/example.jpg) no-repeat top center fixed;
```

It does not matter if one of the values above are missing.

For a complete list of CSS properties, go to **https://www.w3schools.com/cssref/default.asp**

CSS TEXT

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

EXAMPLE	VALUES	WHAT IT DOES
`color: #555555;`	color name, RGB, hexadecimal	*Specifies the color of a text element*
`font-family: oswald, sans-serif;`	family-name, generic-family	*Specifies the font, with fallbacks**
`font-size: 2em;`	px, em, rem, vw, %	*Sets the size of the font*
`font-weight: bold;`	normal, bold, bolder, lighter, 100-900	*Specifies a font weight*
`font-style: italic;`	normal, italic	*Specifies a font style*
`letter-spacing: 0.2em;`	px, em, rem	*Changes space between characters*
`line-height: 1.5;`	number*, px, em, rem	*Specifies the height of a line (like leading)*
`text-align: center;`	left, right, center, justify	*Specifies the horizontal alignment of text*
`text-decoration-line: none;`	none, underline, overline, linethrough	*Sets the kind of text decoration (usu. for links)*
`text-decoration-style: wavy;`	solid, doube, dotted, dashed, wavy	*Sets the style of the text decoration*
`text-indent: 20px;`	px, em, rem	*Specifies indentation of first line in text-block*
`text-transform: uppercase;`	capitalize, uppercase, lowercase	*Controls the capitalization of text*

*A number that will be multiplied with the current font-size to set the line height; a value of 1.5 means 1.5 times the font size

SHORTHAND

You can set several font properties in one declaration, in the following order: font-style, font-weight/line-height, font-size, font-family.

```
font: italic bold 12px/30px georgia, serif;
```

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

SPECIFYING FONTS

The `font-family` property can hold several font names as a “fallback” system. If the browser does not support the first font, it tries the next font. There are two types of font family names: the precise name of a font-family, like “oswald”, or the name of a generic-family. Start with the font you want, and always end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available. There are five generic font families:

Sans-serif	Serif	Monospace	*Cursive*	Fantasy
(Arial)	**(Times)**	**(Courier)**	**(Chancery)**	**(Papyrus)**

CSS LAYOUT

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. A block-level element always starts on a new line and takes up the full width available. An inline element does not start on a new line and only takes up as much width as necessary.

EXAMPLE	VALUES	WHAT IT DOES
`display: block;`	inline, block, inline-block, none	*Specifies the display behavior of an element*
`visibility: hidden;`	visible (default), hidden	*Specifies whether or not an element is visible*
`overflow: hidden;`	visible (default), hidden, scroll, auto	*Specifies what happens if content overflows an element's box*

Use the display property to change an element's default behavior — to make a block-level element display inline, or vice versa. If an element is hidden (using the visibility property), it still takes up space on the page. Use a display property of none to both hide *and* remove an element from the document layout!

EXAMPLE	VALUES	WHAT IT DOES
`width: 80%;`	auto, px, vw, %	*Sets width of an element; percent is relative to containing block*
`height: 300px;`	auto, px, vh, %	*Sets height of an element; percent is relative to containing block*

By default, a block-level element will fill the available width on the browser window, but setting a width value for your content allows you to control its place in the page flow. For example, if we set a width value for our body element to be 1000px, we can set the width for one of its nested elements — like the main — to, say, 80 percent of that 1000px. The height of an element is by default set to its height, but we can control this as well by writing a rule to provide a specific value for the height of any element.

EXAMPLE	VALUES	WHAT IT DOES
`position: relative;`	relative, absolute, fixed, sticky	*Specifies the type of positioning method used for an element*
`top: -30px`	px, vh, %	*Affects the vertical position of a positioned element*
`right: 30px;`	px, vw, %	*Affects the horizontal position of a positioned element*
`bottom: 30px;`	px, vh, %	*Affects the vertical position of a positioned element*
`left: 30px;`	px, vw, %	*Affects the horizontal position of a positioned element*

By default, an element's position is set to "**static**," which means elements render in order, as they appear in the document flow. But there are four ways to alter this: with a position of **relative**, you can use the four positioning values (top, right, bottom, left) to alter where the element appears *from* its normal position, so "left:30px" rule adds 30 pixels to the element's LEFT position.

The absolute and fixed positions do NOT remain in the page flow. Those elements would appear in relation to their first positioned (not static) ancestor element (**absolute**) or in relation to the browser window (**fixed**). They also take on top, right, bottom or left values. Elements with absolute positioning will scroll with the page, but fixed elements will not.

A **sticky** element is positioned based on the user's scroll position, toggling between relative and fixed. It is positioned relative until a given offset position (via a top declaration) is met in the viewport — then it "sticks" in place (like position:fixed).

EXAMPLE	VALUES	WHAT IT DOES
`float: left;`	left, right, none	*Specifies the display behavior of an element*
`clear: both;`	left, right, both	*Specifies where floating elements are not allowed to float*
`z-index: 1000;`	number	*Specifies the stack order of an element*

The purpose of the **float** property is to push a block-level element to the left or right, taking it out of the flow in relation to other block elements. Any text flows around the floated element. (Be sure to set margin values on floated elements so there is offset between them and the text.) Any element will wrap around the float, unless it has been **clear**ed; with a clear:both rule, an element will not render until the floated element ends. If you need to alter the stacking order (the order in which HTML elements are written), use a **z-index**. (An element with a higher stack order is in front of an element with a lower one.) A z-index only works on positioned elements.

CSS EXTRAS

Try some of these CSS rules to add visual distinction to your website.

EXAMPLE	VALUES	WHAT IT DOES
`box-shadow: 5px 5px 5px rgba(0,0,0,0.5);`	px / color	*Adds a shadow to an object*
`text-shadow: 5px 5px 5px rgba(0,0,0,0.5);`	px / color	*Adds a shadow to text*

The values for these shadow effects are in the following order: horizontal offset, vertical offset, amount of blur, color.

EXAMPLE	WHAT IT DOES
`background-image: linear-gradient(to right, red,orange,yellow);`	*Adds a gradient to an object*
`background-image: radial-gradient(circle, red 20%,orange 75%, yellow 90%);`	*Adds a gradient to an object*

These function rules allow you to add a gradient background to an element. The first is linear, left to right, with three color stops. The second is radial, shaped as a circle (allipse is the default) with three color stops at specific locations.

EXAMPLE	VALUES	WHAT IT DOES
`transform: translate(50%,50%);`	px, %	*Defines a 2D translation (moving an element's x and y positions)*
`transform: scale(1.2,1.2);`	number	*Defines a 2D scale translation, scaling x (width) and y (height)*
`transform: rotate(5deg);`	deg	*Defines a 2D rotation translation (in degrees)*

The transform property applies a 2D (or 3D) transformation to an element, allowing you to rotate, scale, move, skew, etc., elements. Try setting these declarations on **:hover** pseudoclasses.

EXAMPLE	VALUES	WHAT IT DOES
`transition-property: width,height;`	property	*Specifies the CSS property the transition effect is for*
`transition-duration: 1s;`	s or ms	*Specifies how long the transition effect takes (seconds or milliseconds)*
`transition-delay: 2s;`	s or ms	*Specifies when the transition effect will start (seconds or milliseconds)*
`transition-timing-function: linear;`	(below)	*Allows a transition effect to change speed over its duration*

The transition property allows a property to change smoothly. Try setting these declarations on **:hover** pseudoclasses.

The timing functions include ease-in (slow start), ease-out (slow end) and ease-in-out (slow start and end).

SHORTHAND

You can set several transition properties in one declaration, in the following order: transition-property, transition-duration, transition-timing-function, transition-delay.

```
transition: width .35s ease-in-out;
```

Property and duration are required; always specify a transition-duration property, otherwise the duration is 0s, and the transition will have no effect.