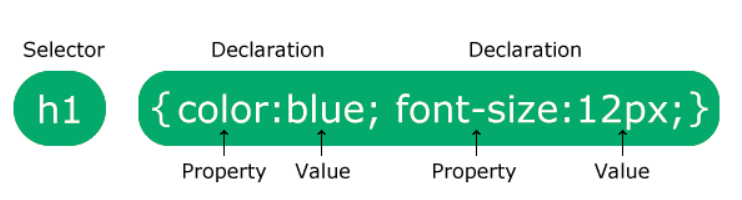
**CSS**

**Css Introduction**

* CSS stands for Cascading Style Sheets
* Used to style HTML document.

**Css Syntax:**

* A CSS rule consists of a selector and a declaration block.
* The selector points to the HTML element you want to style.
* The declaration block contains one or more declarations separated by semicolons.
* Each declaration includes a CSS property name and a value, separated by a colon.
* Declaration blocks are surrounded by curly braces.



* Example:

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

**Explanation**

* p is a selector (it points to the HTML element we want to style: <p>).
* color is a property, and ‘red’ is the property value.
* text-align is a property, and ‘center’ is the property value.

**Css comment:**

* Comments are used to explain the code.
* Comment not interpreted by browser.
* Syntax:

/\* comment \*/

**CSS Selectors**

* CSS selectors are used to select the HTML element.
* CSS selectors categories:
* **Simple selectors** (element, id, class)
* **Combinator selectors** (select elements based on 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)

**Simple selectors**

**Element select:**

* Element selector selects HTML elements based on the element name.
* Example:

p {

text-align: center;

color: red;

}

**Id Selector:**

* Id selector uses the **id attribute** of an HTML element to select a specific element.
* 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.
* Note: An id name cannot start with a number.
* Example:

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

**Class Selector:**

* Selects HTML elements with a specific **class attribute** of HTML element.
* To select elements with a specific class, write a period (.) character, followed by the class name.
* In HTML, Multiple element can have same class name.
* And also one element can have multiple class separated by class.
* **Note:** A class name cannot start with a number.
* Example

.center {

text-align: center;

color: red;

}

* We can also specify that only specific HTML elements should be affected by a class.
* Example:

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

/\* Styled applied to only <p> elements with class="center" \*/

**Universal Selector:**

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

\* {

text-align: center;

color: blue;

}

/\* This style apply to all element in html document \*/

**Grouping Selector:**

* It is used to select multiple element with same style.
* Example:

h1, h2, p {

text-align: center;

color: red;

}

/\* This style apply to **h1**, **h2** and **p** element in html document \*/

**Combinator selectors**

**Descendant Selector (space):**

* The descendant selector matches all elements that are descendants of a specified element.
* Example:

div p {

background-color: yellow;

}

/\* selects all <p> elements inside <div> elements \*/

**Direct Child Selector (>):**

* The child selector selects all elements that are the direct children of a specified element.
* Example:

div > p {

background-color: yellow;

}

**Adjacent Sibling Selector (+)**

* The adjacent sibling selector is used to select an element that is directly after another specific element.
* Sibling elements must have the same parent element, and "adjacent" means "immediately following".
* Example:

div + p {

background-color: yellow;

}

**General Sibling Selector (~)**

* The general sibling selector selects all elements that are next siblings of a specified element.
* Example:

div ~ p {  
  background-color: yellow;  
}

/\* selects all <p> elements that are **next** siblings of <div> elements \*/

**Pseudo-class selectors:**

* A pseudo-class is used to select a specific state of an element.
* Syntax of pseudo-classes:

selector:pseudo-class {

property: value;

}

**Anchor Pseudo-classes**

* Example:

/\* unvisited link \*/

a:link {

color: #FF0000;

}

/\* visited link \*/

a:visited {

color: #00FF00;

}

/\* mouse over link \*/

a:hover {

color: #FF00FF;

}

/\* selected link \*/

a:active {

color: #0000FF;

}

**Note:**

a:hover MUST come after a:link and a:visited in the CSS definition in order to be effective!

a:active MUST come after a:hover in the CSS definition in order to be effective!

**:first-child Pseudo-class**

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

p:first-child {

color: blue;

}

**:lang Pseudo-class**

* The :lang pseudo-class allows you to define special rules for different languages.
* Example:

Html:

<p>

Some text

<q lang="fr">A quote in a paragraph</q>

Some text.

</p>

Style:

q:lang(fr) {

background: red;

}

**Pseudo Classes list**

|  |  |  |
| --- | --- | --- |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | Selects the active link |
| [:disabled](https://www.w3schools.com/cssref/sel_disabled.asp) | input:disabled | Selects every disabled <input> element |
| [:empty](https://www.w3schools.com/cssref/sel_empty.asp) | p:empty | Selects every <p> element that has no children |
| [:enabled](https://www.w3schools.com/cssref/sel_enabled.asp) | input:enabled | Selects every enabled <input> element |
| [:first-child](https://www.w3schools.com/cssref/sel_firstchild.asp) | p:first-child | Selects every <p> elements that is the first child of its parent |
| [:first-of-type](https://www.w3schools.com/cssref/sel_first-of-type.asp) | p:first-of-type | Selects every <p> element that is the first <p> element of its parent |
| [:focus](https://www.w3schools.com/cssref/sel_focus.asp) | input:focus | Selects the <input> element that has focus |
| [:hover](https://www.w3schools.com/cssref/sel_hover.asp) | a:hover | Selects links on mouse over |
| [:in-range](https://www.w3schools.com/cssref/sel_in-range.asp) | input:in-range | Selects <input> elements with a value within a specified range |
| [:invalid](https://www.w3schools.com/cssref/sel_invalid.asp) | input:invalid | Selects all <input> elements with an invalid value |
| [:lang(*language*)](https://www.w3schools.com/cssref/sel_lang.asp) | p:lang(it) | Selects every <p> element with a lang attribute value starting with "it" |
| [:last-child](https://www.w3schools.com/cssref/sel_last-child.asp) | p:last-child | Selects every <p> elements that is the last child of its parent |
| [:last-of-type](https://www.w3schools.com/cssref/sel_last-of-type.asp) | p:last-of-type | Selects every <p> element that is the last <p> element of its parent |
| [:link](https://www.w3schools.com/cssref/sel_link.asp) | a:link | Selects all unvisited links |
| [:nth-child(n)](https://www.w3schools.com/cssref/sel_nth-child.asp) | p:nth-child(2) | Selects every <p> element that is the second child of its parent |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | Selects <input> elements with no "required" attribute |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | Selects <input> elements with a "readonly" attribute specified |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | Selects <input> elements with a "required" attribute specified |
| [:valid](https://www.w3schools.com/cssref/sel_valid.asp) | input:valid | Selects all <input> elements with a valid value |
| [:visited](https://www.w3schools.com/cssref/sel_visited.asp) | a:visited | Selects all visited links |

**Pseudo-element selectors:**

* A CSS pseudo-element is used to style specified parts of an element.
* Syntax:

selector::pseudo-element {  
  property: value;  
}

**::first-line Pseudo-element:**

* The ::first-line pseudo-element is used to add a special style to the first line of a text.
* The ::first-line pseudo-element can only be applied to block-level elements.
* Example:

p::first-line {

color: #ff0000;

}

**::first-letter Pseudo-element:**

* The ::first-letter pseudo-element is used to add a special style to the first letter of a text.
* The ::first-letter pseudo-element can only be applied to block-level elements.
* Example:

p::first-letter {

color: #ff0000;

font-size: xx-large;

}

**::before Pseudo-element:**

* The ::before pseudo-element can be used to insert some content before the content of an element.
* Example:

p::before {

content: "Before Psuedo Element";

color: lightblue;

font-size: 2rem;

}

**::after Pseudo-element:**

* The ::after pseudo-element can be used to insert some content after the content of an element.
* Example:

p::after {

content: "After Psuedo Element";

color: lightblue;

font-size: 2rem;

}

**::marker Pseudo-element**

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

::marker {

color: red;

font-size: 23px;

}

**::selection Pseudo-element**

* The ::selection pseudo-element matches the portion of an element that is selected by a user.
* Example:

::selection {

background-color: lightblue;

color: black;

}

**Attribute selectors:**

* The attribute selector is used to select elements with a specified attribute.

**[attribute]:**

* Used to select elements with a specified attribute.
* Example:

a[target] {

background-color: yellow;

}

**[attribute=value]:**

* Used to select elements with a specified attribute and value.
* Example:

a[target="\_blank"] {

background-color: yellow;

}

**[attribute~=value]**

* Used to select elements with an attribute value containing a specified word.
* Example:

[title~="flower"] {

border: 5px solid yellow;

}

**[attribute|=value]**

* Used to select elements with the specified attribute, whose value can be exactly the specified value, or the specified value followed by a hyphen (-)
* Example:

[class|="top"] {

background: yellow;

}

**[attribute^=value]**

* Used to select elements with the specified attribute, whose value starts with the specified value.
* Example:

[class^="top"] {

background: yellow;

}

**[attribute$="value"]**

* Used to select elements whose attribute value ends with a specified value.
* Example:

[class$="test"] {

background: yellow;

}

**[attribute\*=value]**

* Used to select elements whose attribute value contains a specified value.
* The value does not have to be a whole word.
* Example:

[class\*="te"] {

background: yellow;

}

**Colors:**

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

**RGB:**

* RGB color value represents Red, Green and Blue
* The value of Red, Green and Blue is between 0 to 255.

rgb(red, green, blue)

**RGBA:**

* RGBA color values are an extension of RGB color values with an alpha channel.
* alpha specifies the opacity for a color.
* The value of alpha if between 0.0 (fully transparent) and 1.0 (not transparent at all).

rgba(red, green, blue, alpha)

**Hex:**

* A hexadecimal color is specified with: #RRGGBB,

where the RR (red), GG (green) and BB (blue) hexadecimal integers specify the components of the color.

* RR (red), GG (green) and BB (blue) are hexadecimal values between 00 and ff.

**3 Digit HEX:**

* The 3-digit hex code is a short form for some 6-digit hex codes.
* It specified with: #RGB
* Value of R, G and B is between 0 to f.

**HSL:**

* HSL stands for hue, saturation, and lightness.
* The value of hue is between 0 to 360. 0 is red, 120 is green, and 240 is blue.
* The value of saturation is between 0% to 100%. 0% means a shade of gray, and 100% is the full color.
* The value of lightness is between 0% to 100%. 0% is black and 100% is white.

hsla(hue, saturation, lightness, alpha)

**HSLA:**

* HSLA color values are an extension of HSL color values with an alpha channel
* alpha specifies the opacity for a color.
* The value of alpha is between 0 to 1.

hsla(hue, saturation, lightness, alpha)

**Backgrounds:**

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

**Background color:**

* background-color property specifies the background color of an element.
* Example:

body {

background-color: lightblue;

}

**Opacity:**

* The opacity property specifies the opacity/transparency of an element.
* It can take a value from 0.0 - 1.0.
* Example:

div {

background-color: rgba(0,255,0,0.5);

/\* opacity: 0.3; \*/

}

**Background image:**

* Used to specify the background image of an element.
* Example:

body {

background-image: url("1.png"), url(‘2.png’), url(‘3.png’), url(‘4.png’), linear-gradient(to top right, red, blue);

background-position: top-left, top-right, bottom-left, bottom-right, center;

background-size: 100px, 100px, 100px, 100px, conver;

}

* To specify multiple background image use multiple url(‘’) separated by comma.
* The first image specified by url(‘’), come Top of the other image specified by other url(‘’).
* Example:

.multiple-img {

background-image: url(img\_flwr.gif), url(paper.gif);

background-position: right bottom, left top;

background-repeat: no-repeat, repeat;

padding: 15px;

}

**Background repeat:**

* By default, the background-image property repeats an image both horizontally and vertically.
* background-repeat : repeat

: repeat-x;

: repeat-y

: no-repeat

**Background position:**

* The background-position property is used to specify the position of the background image.
* Background-position : x-position y-postion;

**x-position** -> left, right, center, %, px, etc..

**y-postion** -> top, bottom, center, %, px, etc..

**Background attachment:**

* The background-attachment property specifies whether the background image should scroll or be fixed while scrolling the page.
* background-attachment : fixed;

: scroll;

: local;

**Background size:**

* The background-size property specifies the size of the background images.
* background-size : auto

: length

: x-len y-len

: cover

: contain

**Background clip:**

* The background-clip property defines how far the background (color or image) should extend within an element.
* background-clip : border-box

: padding-box

: content-box

**Background origin:**

* The background-origin property specifies the origin position (the background positioning area) of a background image.
* Note: This property has no effect if background-attachment is "fixed" or background-repeat is “repeat”.
* Possible values are content-box, border-box, padding-box.
* Example:

.background-origin {

border: 10px dashed black;

padding: 25px;

background: url(paper.gif);

background-repeat: no-repeat;

background-origin: content-box;

}

**Background Shorthand:**

* It is short handle property for background properties,
* Syntax:

background: bg-color bg-image position bg-size bg-repeat bg-origin bg-clip bg-attachment | initial | inherit;

**Border:**

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

**Border Style:**

* The border-style property specifies what kind of border to display.
* The following values are allowed:

dotted - Defines a dotted border

dashed - Defines a dashed border

solid - Defines a solid border

double - Defines a double border

groove - Defines a 3D grooved border. The effect depends on the border-color value

ridge - Defines a 3D ridged border. The effect depends on the border-color value

inset - Defines a 3D inset border. The effect depends on the border-color value

outset - Defines a 3D outset border. The effect depends on the border-color value

none - Defines no border

hidden - Defines a hidden border

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

* Example:

p.solid {

border-style: solid;

}

p.mix {

border-style: dotted dashed solid double;

}

**Border Width:**

* The border-width property specifies the width of the four borders.
* The width can be set as a specific size (in px, pt, cm, em, etc) or by using one of the three pre-defined values: **thin**, **medium(default)**, or **thick.**
* Example

.border-1 {

border-style: solid;

border-width: 5px;

}

.border2 {

border-style: solid;

border-width: medium;

}

**Border Color:**

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

.border-color {

border-style: solid;

border-color: red;

}

**Border Shorthand Property:**

* The border property is a shorthand property for the following individual border properties:

border-width

border-style (required)

border-color

* Example:

p {

border: 1px solid red;

}

**Rounded Borders:**

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

p {

border: 2px solid red;

border-radius: 5px;

}

* **border-radius: 15px 50px 30px 5px;** (first value applies to **top-left** corner, second value applies to **top-right** corner, third value applies to **bottom-right** corner, and fourth value applies to **bottom-left** corner)
* **border-radius: 15px 50px 30px;** (first value applies to **top-left** corner, second value applies to **top-right** and **bottom-left** corners, and third value applies to **bottom-right** corner)
* **border-radius: 15px 50px;** (first value applies to **top-left** and **bottom-right** corners, and the second value applies to **top-right** and **bottom-left** corners)

**Border Image**

**border-image-source:**

* The border-image-source CSS property sets the source image used to create an element's border image.
* Syntax:

border-image-source: url(image.jpg);

border-image-source: linear-gradient(to top, red, yellow);

border-image-source: radial-gradient(red, yellow);

border-image-source: conic-gradient(red, yellow);

**border-image-width:**

* The border-image-width property specifies the width of the border image.
* Syntax

/\* number value \*/

border-image-width: 3;

/\* top and bottom | left and right \*/

border-image-width: 2em 3em;

/\* top | left and right | bottom \*/

border-image-width: 5% 15% 10%;

/\* top | right | bottom | left \*/

border-image-width: 5% 2em 10% auto;

**border-image-slice:**

* The slicing process creates nine regions in total: four **corners**, four **edges**, and a **middle** region.
* Four slice lines, set a given distance from their respective sides, control the size of the regions.
* Syntax:

/\* All sides \*/

border-image-slice: 30%;

/\* top and bottom | left and right \*/

border-image-slice: 10% 30%;

/\* top | left and right | bottom \*/

border-image-slice: 30 30% 45;

/\* top | right | bottom | left \*/

border-image-slice: 7 12 14 5;

**border-image-** **repeat:**

* The border-image-repeat CSS property defines how the edge regions and middle region of a source image are adjusted to fit the dimensions of an element's border image.
* The middle region can be displayed by using the keyword "fill" in the border-image-slice property.
* Syntax:

border-image-repeat: stretch;

border-image-repeat: repeat;

border-image-repeat: round;

border-image-repeat: space;

**border-image-** **outset:**

* The border-image-outset CSS property sets the distance by which an element's border image is set out from its border box.
* Syntax:

/\* <number> value \*/

border-image-outset: 1.5;

/\* top and bottom | left and right \*/

border-image-outset: 1 1.2;

/\* top | left and right | bottom \*/

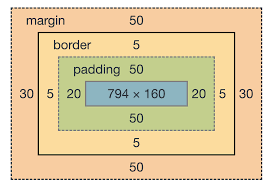
border-image-outset: 30px 2 45px;

/\* top | right | bottom | left \*/

border-image-outset: 7px 12px 14px 5px;

**Box Model:**

* All HTML elements can be considered as boxes.
* The CSS box model is essentially a box that wraps around every HTML element. It consists of: margins, borders, padding, and the actual content.



**Width and Height of an Element**

* When you set the width and height properties of an element with CSS, you just set the width and height of the content area. To calculate the full size of an element, you must also add padding, borders and margins.
* Example:

div {

width: 320px;

padding: 10px;

border: 5px solid gray;

margin: 20;

}

Total Width = 320px (content width)

+ 10px + 10px (left + right padding)

+ 5px + 5px (left + right border)

+ 20px + 20px (left + right margin)

= 390px

**Total element width** = width + left padding + right padding + left border + right border + left margin + right margin

**Total element height** = height + top padding + bottom padding + top border + bottom border + top margin + bottom margin

**Box Sizing:**

* box-sizing property allows us to include the padding and border in an element's width and height.
* If we set box-sizing: border-box; on an element, padding and border are included in the width and height:

**Total element width** = width + left margin + right margin

**Total element height** = height + top margin + bottom margin

* box-sizing : context-box;

: border-box;

**Margins:**

* Margins are used to create space around elements, outside of any defined borders.
* Properties for specifying the margin for each side of an element:

margin-top

margin-right

margin-bottom

margin-left

* Shorthand Property:

margin: top right bottom left;

margin: top rightleft bottom;

margin: topbottom rightleft;

* Example:

p {

margin-top: 100px;

margin-bottom: 100px;

margin-right: 150px;

margin-left: 80px;

}

* Example:

p {

margin: 25px 50px 75px 100px;

}

* Example:

p {

margin: 25px 50px 75px;

}

**Margins Collapse:**

* Top and bottom margins of elements are sometimes collapsed into a single margin that is equal to the largest of the two margins.
* This does not happen on left and right margins! Only top and bottom margins.
* Example:

h1 {

margin: 0 0 50px 0;

}

h2 {

margin: 20px 0 0 0;

}

**Paddings:**

* Padding is used to create space around an element's content, inside of any defined borders.
* Padding can not have negative value (treated as 0).
* Properties for specifying the padding for each side of an element:

padding-top

padding-right

padding-bottom

padding-left

* Shorthand Property:

padding: top right bottom left;

padding: top rightleft bottom;

padding: topbottom rightleft;

* Example:

div {

padding-top: 50px;

padding-right: 30px;

padding-bottom: 50px;

padding-left: 80px;

}

* Example:

div {

padding: 25px 50px 75px 100px;

}

**Outline:**

* An outline is a line drawn outside the element's **border**.
* Border vs outline

**Border:**

* A border is used to define the boundaries of an element.
* It is a part of the box model in CSS and can be set using properties like border-width, border-style, and border-color. Borders are typically applied to the edges of an element, and they can be solid, dashed, dotted, etc.
* Borders are an integral part of the layout and design of a webpage and are used to separate and visually distinguish elements.

**Outline:**

* An outline, on the other hand, is similar to a border in that it creates a visual boundary around an element.
* However, outlines are drawn outside the border edge, and they **don't take up space** or affect the layout of the element or surrounding elements. The outline is used mainly for visual emphasis or to indicate focus (such as when navigating a webpage using the keyboard).
* The outline property in CSS allows you to set the style, color, and width of an element's outline.
* It's commonly used in cases where you want to **highlight an element** without affecting its layout. For example, when you click on an input field, it might get an outline to indicate that it is currently focused.
* CSS has the following outline properties:

outline-style

outline-color

outline-width

outline-offset

outline

**Outline style:**

* The outline-style property specifies the style of the outline, and can have one of the following values:

dotted - Defines a dotted outline

dashed - Defines a dashed outline

solid - Defines a solid outline

double - Defines a double outline

groove - Defines a 3D grooved outline

ridge - Defines a 3D ridged outline

inset - Defines a 3D inset outline

outset - Defines a 3D outset outline

none - Defines no outline

hidden - Defines a hidden outline

* Note: None of the other outline properties will have ANY effect unless the outline-style property is set!

**Outline Width:**

* The outline-width property specifies the width of the outline, and can have one of the following values:

thin (typically 1px)

medium (typically 3px)(default)

thick (typically 5px)

A specific size (in px, pt, cm, em, etc)

**Outline Color:**

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

p {

outline-style: solid;

outline-color: red;

}

**Outline Offset:**

* The outline-offset property adds space between an outline and the border of an element.
* Example:

p {

margin: 30px;

border: 1px solid black;

outline: 1px solid red;

outline-offset: 15px;

}

**Outline Shorthand:**

* The outline property is a shorthand property for setting the following individual outline properties:

outline-width

outline-style (required)

outline-color

**Units:**

* CSS has several different units for expressing a length.
* There are two types of length units: absolute and relative.

**Absolute Unit:**

* The absolute length units are fixed and a length expressed in any of these will appear as exactly that size.

|  |  |
| --- | --- |
| cm | Centimetres (1cm = 37.8px) |
| mm | Millimetres (1mm = 3.78px) |
| Q | Quarter millimetres (1Q = ¼cm) |
| In | inches (1in = 96px) |
| px \* | Pixels |
| pt | points (1pt = 1.33px) |
| pc | picas (1pc = 16px) |

**Relative Unit:**

* Relative length units specify a length relative to another length property.

|  |  |
| --- | --- |
| em | Relative to the font-size of the parent element (2em means 2 times the size of the current font of parent). |
| ex | x-height of the parent element's font. |
| ch | Relative to width of the "0" (zero) of parent element. |
| rem | Relative to font-size of the root element. |
| lh | Line height of the parent element. |
| rlh | Line height of the root element. |
| vw | Relative to 1% of the width of the viewport |
| vh | Relative to 1% of the height of the viewport |
| vmin | Relative to 1% of viewport's smaller dimension |
| vmax | Relative to 1% of viewport's larger dimension |
| % | Relative to the parent element |

* Viewport is the browser window size.
* There are also other units like vb, vi, svw, svh, lvh, lvw, dvw, dvh etc...