Cascading Sytle Sheets (CSS)

1. CSS is a language that describes the style of an HTML document. CSS describes how HTML elements should be displayed.
2. CSS is used to define styles for your web pages, including the design, layout and variations in display for different devices and screen sizes

CSS: A CSS rule-set consist of selector and decoration block. The selector points to the HTML element you want to style.

The element selector: It selects elements based on the element name in HTML.

The Id selector: It used the id attribute of an HTML element to select a specific element in HTML. To select an element with a specific id, write a hash (#) character, followed by the id of the element.

The Class selector: It selects elements with a specific class attribute. To select elements with a specific class, write a period (.) character, followed by the name of the class.

Grouping Selector: If you have elements with the same style definitions, it will be better to group the selectors, to minimize the code. To group selectors, separate each selector with a comma.

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

There are three ways of inserting a style sheet: External (in some file) .css file , Internal (inside style tag), inline (within the body of the html)

## Multiple Style Sheets

If some properties have been defined for the same selector (element) in different style sheets, the value from the last read style sheet will be used.

## Cascading Order

What style will be used when there is more than one style specified for an HTML element?

Generally speaking, we can say that all the styles will "cascade" into a new "virtual" style sheet by the following rules, where number one has the highest priority:

1. Inline style (inside an HTML element)
2. External and internal style sheets (in the head section)
3. Browser default

Color in CSS is not case-sensitive like RED or red or Red all are same.

CSS background properties:

* background-color: The background-color property specifies the background color of an element.
* 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. By default, the background-image property repeats an image both horizontally and vertically.
* background-repeat: **repeat-x** (repeated only horizontally), **repeat-y** (repeated only vertically), **no-repeat** (there will be no repeat)
* background-attachment: **fixed -** To specify that the background image should be fixed (will not scroll with the rest of the page)
* background-position: **right top** (We want to change the position of the image, so that it does not disturb the text too much.)
* 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. The shorthand property for background is background.

When using the shorthand property, the order of the property values is:

* background-color
* background-image
* background-repeat
* background-attachment
* background-position

CSS border properties:

The border-style property specifies what kind of border to display. The border-style property can have from one to four values (for the top border, right border, bottom border, and the left border).

The border-width property specifies the width of the four borders. The border-width property can have from one to four values (for the top border, right border, bottom border, and the left border).

he border-color property is used to set the color of the four borders. The border-color property can have from one to four values (for the top border, right border, bottom border, and the left border). If border-color is not set, it inherits the color of the element.

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

* border-width
* border-style (required)
* border-color

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

CSS Margin :

The CSS margin properties are used to create space around elements, outside of any defined borders.

CSS has properties for specifying the margin for each side of an element:

* margin-top
* margin-right
* margin-bottom
* margin-left

All the margin properties can have the following values:

* auto - the browser calculates the margin
* length - specifies a margin in px, pt, cm, etc.
* % - specifies a margin in % of the width of the containing element
* inherit - specifies that the margin should be inherited from the parent element

The margin property is a shorthand property for the following individual margin properties:

* margin-top
* margin-right
* margin-bottom
* margin-left

Margin Auto: You can set the margin property to auto to horizontally center the element within its container. The element will then take up the specified width, and the remaining space will be split equally between the left and right margins:

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

Margin 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!

## CSS Padding:

## The CSS padding properties are used to generate space around an element's content, inside of any defined borders. The padding property is a shorthand property for the individual padding properties in order top,right,bottom and left

All the padding properties can have the following values:

* length - specifies a padding in px, pt, cm, etc.
* % - specifies a padding in % of the width of the containing element
* inherit - specifies that the padding should be inherited from the parent element

The CSS width property specifies the width of the element's content area. The content area is the portion inside the padding, border, and margin of an element ([the box model](https://www.w3schools.com/css/css_boxmodel.asp)). So, if an element has a specified width, the padding added to that element will be added to the total width of the element. This is often an undesirable result. In the following example, the <div> element is given a width of 300px. However, the actual rendered width of the <div> element will be 350px (300px + 25px of left padding + 25px of right padding):

To keep the width at 300px, no matter the amount of padding, you can use the box-sizing property. This causes the element to maintain its width; if you increase the padding, the available content space will decrease. Here is an example:

div.ex2 { width: 300px; padding: 20px; box-sizing: border-box; background-color: lightblue; }

CSS Height & Width:

The height and width properties are used to set the height and width of an element. The height and width can be set to auto (this is default. Means that the browser calculates the height and width), or be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block.

The height and width properties do not include padding, borders, or margins; they set the height/width of the area inside the padding, border, and margin of the element!

Max-width: The max-width property is used to set the maximum width of an element. The max-width can be specified in *length values*, like px, cm, etc., or in percent (%) of the containing block, or set to none (this is default. Means that there is no maximum width).

The problem with the <div> above occurs when the browser window is smaller than the width of the element (500px). The browser then adds a horizontal scrollbar to the page.Using max-width instead, in this situation, will improve the browser's handling of small windows.

**Tip:** Drag the browser window to smaller than 500px wide, to see the difference between the two divs!

**CSS Bos Model:**

All HTML elements can be considered as boxes. 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:

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

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

The total width of an element should be calculated like this:

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

The total height of an element should be calculated like this:

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

CSS Outline:

An outline is a line that is drawn around elements, OUTSIDE the borders, to make the element "stand out". It works same as border property of CSS. It contain outline-offset properties CSS has the following outline properties:

* outline-style
* outline-color
* outline-width
* outline-offset: The outline-offset property adds space between an outline and the edge/border of an element. The space between an element and its outline is transparent.
* outline

**Note**: Outline differs from borders! Unlike 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 Text:

The text-align property is used to set the horizontal alignment of a text. A text can be left or right aligned, centered, or justified. The following example shows center aligned, and left and right aligned text (left alignment is default if text direction is left-to-right, and right alignment is default if text direction is right-to-left):

When the text-align property is set to "justify", each line is stretched so that every line has equal width, and the left and right margins are straight (like in magazines and newspapers):

The text-decoration property is used to set or remove decorations from text. The value text-decoration: none; is often used to remove underlines from links: We have other text-decoration value overline, line-through, underline.

The text-transform property is used to specify uppercase and lowercase letters in a text. It can be used to turn everything into uppercase or lowercase letters, or capitalize the first letter of each word:

The text-indent property is used to specify the indentation of the first line of a text:

The letter-spacing property is used to specify the space between the characters in a text.The following example demonstrates how to increase or decrease the space between characters:

For eg : letter-spacing : 3px or letter-spacing:-3px

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

The direction property is used to change the text direction of an element:

The word-spacing property is used to specify the space between the words in a text.

The text-shadow property adds shadow to text.The following example specifies the position of the horizontal shadow (3px), the position of the vertical shadow (2px) and the color of the shadow (red).

The [text-overflow](https://www.w3schools.com/cssref/css3_pr_text-overflow.asp) Specifies how overflowed content that is not displayed should be signaled to the user.

The [vertical-align](https://www.w3schools.com/cssref/pr_pos_vertical-align.asp) Sets the vertical alignment of an element.

CSS Font-family:

The font family of a text is set with the font-family property.The font-family property should hold several font names as a "fallback" system. If the browser does not support the first font, it tries the next font, and so on.Start with the font you want, and end with a generic family, to let the browser pick a similar font in the generic family, if no other fonts are available.

**Note**: If the name of a font family is more than one word, it must be in quotation marks, like: "Times New Roman".More than one font family is specified in a comma-separated list:

Font-style:

The font-style property is mostly used to specify italic text. This property has three values:

* normal - The text is shown normally
* italic - The text is shown in italics
* oblique - The text is "leaning" (oblique is very similar to italic, but less supported)

Font-size:

The font-size property sets the size of the text. Being able to manage the text size is important in web design. However, you should not use font size adjustments to make paragraphs look like headings, or headings look like paragraphs. Always use the proper HTML tags, like <h1> - <h6> for headings and <p> for paragraphs.

The font-size value can be an absolute, or relative size.

Absolute size:

* Sets the text to a specified size
* Does not allow a user to change the text size in all browsers (bad for accessibility reasons)
* Absolute size is useful when the physical size of the output is known

Relative size:

* Sets the size relative to surrounding elements
* Allows a user to change the text size in browsers

**Note:** If you do not specify a font size, the default size for normal text, like paragraphs, is 16px (16px=1em).

The font-variant property specifies whether or not a text should be displayed in a small-caps font. In a small-caps font, all lowercase letters are converted to uppercase letters. However, the converted uppercase letters appear in a smaller font size than the original uppercase letters in the text.

The font shorthand property sets all the font properties in one declaration. The properties that can be set, are (in order): "font-style font-variant font-weight font-size/line-height font-family"

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>).

CSS links:

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

When setting the style for several link states, there are some order rules:

* a:hover MUST come after a:link and a:visited
* a:active MUST come after a:hover

CSS List Properties:

In HTML, there are two main types of lists:

* unordered lists (<ul>) - the list items are marked with bullets
* ordered lists (<ol>) - the list items are marked with numbers or letters

The CSS list properties allow you to:

* Set different list item markers for ordered lists
* Set different list item markers for unordered lists
* Set an image as the list item marker
* Add background colors to lists and list items

The list-style-type property specifies the type of list item marker. It has values circle, square, upper-roman & lower-alpha

The list-style-image property specifies an image as the list item marker:

The list-style-position property specifies whether the list-item markers should appear inside or outside the content flow:

The list-style-type: none property can also be used to remove the markers/bullets. Note that the list also has default margin and padding. To remove this, add margin:0 and padding:0 to <ul> or <ol>:

The list-style property is a shorthand property. the order of the property values are:

* list-style-type (if a list-style-image is specified, the value of this property will be displayed if the image for some reason cannot be displayed)
* list-style-position (specifies whether the list-item markers should appear inside or outside the content flow)
* list-style-image (specifies an image as the list item marker)

CSS Display Properties:

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.

## Block-level Elements: A block-level element always starts on a new line and takes up the full width available (stretches out to the left and right as far as it can). The <div> element is a block-level element. Examples of block-level elements:

* <div>
* <h1> - <h6>
* <ul><ol>
* <p>
* <form>
* <header>
* <footer>
* <section>

**Inline Elements**: An inline element does not start on a new line and only takes up as much width as necessary. This is an inline <span> element inside a paragraph. Examples of inline elements:

* <span>
* <a>
* <img>

display: none; is commonly used with JavaScript to hide and show elements without deleting and recreating them. Take a look at our last example on this page if you want to know how this can be achieved. The <script> element uses display: none; as default

**Note**: Setting the display property of an element only changes **how the element is displayed**, NOT what kind of element it is. So, an inline element with display: block; is not allowed to have other block elements inside it.

Hiding an element can be done by setting the display property to none. The element will be hidden, and the page will be displayed as if the element is not there:

visibility:hidden; also hides an element. However, the element will still take up the same space as before. The element will be hidden, but still affect the layout:

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

**Static**: HTML elements are positioned static by default. Static positioned elements are not affected by the top, bottom, left, and right properties. An element with position: static; is not positioned in any special way; it is always positioned according to the normal flow of the page:

position need to be understand more . That we will do later

**CSS Z-index**: When elements are positioned, they can overlap other elements. The z-index property specifies the stack order of an element (which element should be placed in front of, or behind, the others).An element can have a positive or negative stack order: An element with greater stack order is always in front of an element with a lower stack order.

**Note:** If two positioned elements overlap without a z-index specified, the element positioned last in the HTML code will be shown on top.

CSS Overflow: The overflow property specifies whether to clip content or to add scrollbars when the content of an element is too big to fit in a specified area.

The overflow property has the following values:

* visible - By default, the overflow is visible, meaning that it is not clipped and it renders outside the element's box:
* hidden - With the hidden value, the overflow is clipped, and the rest of the content is hidden:
* scroll - Setting the value to scroll, the overflow is clipped and a scrollbar is added to scroll inside the box. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it):
* auto - The auto value is similar to scroll, only it add scrollbars when necessary:

**Note:** The overflow property only works for block elements with a specified height.

CSS Float: The CSS float property specifies how an element should float. The float property is used for positioning and layout on web pages. The float property can have one of the following values:

* left - The element floats to the left of its container. In this, if there are multiple div in the form of block and if this property is applied then the elements are arranging sequentially from left to right. For eg: if there are multiple image then by default they will arrange from top to bottom and after applying this property this image will arrange sequentially from left to right.
* right- The element floats to the right of its container. Same as left, but the elements order will change, in this case it will be right to left.
* none - The element does not float (will be displayed just where it occurs in the text). This is default
* inherit - The element inherits the float value of its parent

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

CSS Clear: The clear property lets specifies what elements can float beside the cleared element and on which side. The clear property can have one of the following values:

* none - Allows floating elements on both sides. This is default
* left - No floating elements allowed on the left side
* right- No floating elements allowed on the right side
* both - No floating elements allowed on either the left or the right side
* inherit - The element inherits the clear value of its parent

The most common way to use the clear property is after you have used a float property on an element.

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

The following example clears the float to the left. Means that no floating elements are allowed on the left side (of the div):

Clear need to be understand more. That we will do later.

CSS Combinator: A combinator is something that explains the relationship between the selectors. A CSS selector can contain more than one simple selector. Between the simple selectors, we can include a combinator. There are four different combinators in CSS3:

* descendant selector (space): The descendant selector matches all elements that are descendants of a specified element.
* child selector (>): The child selector selects all elements that are the immediate children of a specified element.
* adjacent sibling selector (+): The adjacent sibling selector selects all elements that are the adjacent siblings of a specified element. Sibling elements must have the same parent element, and "adjacent" means "immediately following".
* general sibling selector (~): The general sibling selector selects all elements that are siblings of a specified element.

CSS Pseudo-classes: A pseudo-class is used to define a special state of an element.

For example, it can be used to:

* Style an element when a user mouse over it
* Style visited and unvisited links differently
* Style an element when it gets focus

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! Pseudo-class names are not case-sensitive.

An example of using the :hover pseudo-class on a <div> element: eg : div:hover {  background-color: blue; }.

The :first-child pseudo-class matches a specified element that is the first child of another element. In the following example, the selector matches any <p> element that is the first child of any element:

In the following example, the selector matches the first <i> element in all <p> elements: eg: p i:first-child { color: blue; }

In the following example, the selector matches all <i> elements in <p> elements that are the first child of another element: eg : p:first-child i { color: blue; }

All CSS Pseudo Classes

|  |  |  |
| --- | --- | --- |
| Selector | Example | Example description |
| [:active](https://www.w3schools.com/cssref/sel_active.asp) | a:active | Selects the active link |
| [:checked](https://www.w3schools.com/cssref/sel_checked.asp) | input:checked | Selects every checked <input> element |
| [: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 |
| [:not(selector)](https://www.w3schools.com/cssref/sel_not.asp) | :not(p) | Selects every element that is not a <p> element |
| [: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 |
| [:nth-last-child(n)](https://www.w3schools.com/cssref/sel_nth-last-child.asp) | p:nth-last-child(2) | Selects every <p> element that is the second child of its parent, counting from the last child |
| [:nth-last-of-type(n)](https://www.w3schools.com/cssref/sel_nth-last-of-type.asp) | p:nth-last-of-type(2) | Selects every <p> element that is the second <p> element of its parent, counting from the last child |
| [:nth-of-type(n)](https://www.w3schools.com/cssref/sel_nth-of-type.asp) | p:nth-of-type(2) | Selects every <p> element that is the second <p> element of its parent |
| [:only-of-type](https://www.w3schools.com/cssref/sel_only-of-type.asp) | p:only-of-type | Selects every <p> element that is the only <p> element of its parent |
| [:only-child](https://www.w3schools.com/cssref/sel_only-child.asp) | p:only-child | Selects every <p> element that is the only child of its parent |
| [:optional](https://www.w3schools.com/cssref/sel_optional.asp) | input:optional | Selects <input> elements with no "required" attribute |
| [:out-of-range](https://www.w3schools.com/cssref/sel_out-of-range.asp) | input:out-of-range | Selects <input> elements with a value outside a specified range |
| [:read-only](https://www.w3schools.com/cssref/sel_read-only.asp) | input:read-only | Selects <input> elements with a "readonly" attribute specified |
| [:read-write](https://www.w3schools.com/cssref/sel_read-write.asp) | input:read-write | Selects <input> elements with no "readonly" attribute |
| [:required](https://www.w3schools.com/cssref/sel_required.asp) | input:required | Selects <input> elements with a "required" attribute specified |
| [:root](https://www.w3schools.com/cssref/sel_root.asp) | root | Selects the document's root element |
| [:target](https://www.w3schools.com/cssref/sel_target.asp) | #news:target | Selects the current active #news element (clicked on a URL containing that anchor name) |
| [: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 |

All CSS Pseudo Elements

|  |  |  |
| --- | --- | --- |
| Selector | Example | Example description |
| [::after](https://www.w3schools.com/cssref/sel_after.asp) | p::after | Insert content after every <p> element |
| [::before](https://www.w3schools.com/cssref/sel_before.asp) | p::before | Insert content before every <p> element |
| [::first-letter](https://www.w3schools.com/cssref/sel_firstletter.asp) | p::first-letter | Selects the first letter of every <p> element |
| [::first-line](https://www.w3schools.com/cssref/sel_firstline.asp) | p::first-line | Selects the first line of every <p> element |
| [::selection](https://www.w3schools.com/cssref/sel_selection.asp) | p::selection | Selects the portion of an element that is selected by a user |

CSS Pseudo-elements: A CSS pseudo-element is used to style specified parts of an element. For example, it can be used to:

* Style the first letter, or line, of an element
* Insert content before, or after, the content of an element

**Notice the double colon notation -** ::first-line versus :first-line   
The double colon replaced the single-colon notation for pseudo-elements in CSS3. This was an attempt from W3C to distinguish between **pseudo-classes** and **pseudo-elements**.

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.

When using the opacity property to add transparency to the background of an element, all its child elements inherit the same transparency. This can make the text inside a fully transparent element hard to read: If you do not want to apply opacity to child elements, like in our example above, use **RGBA** color values.

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.

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

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

## CSS [attribute="value"] Selector: The [attribute="value"] selector is used to select elements with a specified attribute and value. The following example selects all <a> elements with a target="\_blank" attribute:

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

## CSS [attribute~="value"] Selector: The [attribute~="value"] selector is used to select elements with an attribute value containing a specified word. The following example selects all elements with a title attribute that contains a space-separated list of words, one of which is "flower":

Eg: [title~="flower"] {border: 5px solid yellow; }

The example above will match elements with title="flower", title="summer flower", and title="flower new", but not title="my-flower" or title="flowers".

## CSS [attribute|="value"] Selector: The [attribute|="value"] selector is used to select elements with the specified attribute starting with the specified value. The following example selects all elements with a class attribute value that begins with "top":

**Note:** The value has to be a whole word, either alone, like class="top", or followed by a hyphen( - ), like class="top-text"!

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

## CSS [attribute^="value"] Selector: The [attribute^="value"] selector is used to select elements whose attribute value begins with a specified value. The following example selects all elements with a class attribute value that begins with "top":

## Note: The value does not have to be a whole word!

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

## CSS [attribute$="value"] Selector: The [attribute$="value"] selector is used to select elements whose attribute value ends with a specified value. The following example selects all elements with a class attribute value that ends with "test":

**Note:** The value does not have to be a whole word!

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

## CSS [attribute\*="value"] Selector: The [attribute\*="value"] selector is used to select elements whose attribute value contains a specified value. The following example selects all elements with a class attribute value that contains "te":

**Note:** The value does not have to be a whole word!

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

CSS Counter: CSS counters are like "variables". The variable values can be incremented by CSS rules (which will track how many times they are used).

To work with CSS counters, we will use the following properties:

* counter-reset - Creates or resets a counter
* counter-increment - Increments a counter value
* content - Inserts generated content
* counter () or counters () function - Adds the value of a counter to an element

To use a CSS counter, it must first be created with counter-reset.

## CSS3 Modules

CSS3 has been split into "modules". It contains the "old CSS specification" (which has been split into smaller pieces). In addition, new modules are added.

Some of the most important CSS3 modules are:

* Selectors
* Box Model
* Backgrounds and Borders
* Image Values and Replaced Content
* Text Effects
* 2D/3D Transformations
* Animations
* Multiple Column Layout
* User Interface

Most of the new CSS3 properties are implemented in modern browsers.

CSS border-radius with Four values: first value applies to top-left, second value applies to top-right, third value applies to bottom-right, and fourth value applies to bottom-left corner

**CSS border-image**: The CSS3 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:

1. The image to use as the border
2. Where to slice the image
3. Define whether the middle sections should be repeated or stretched

The border-image property takes the image and slices it into nine sections, like a tic-tac-toe board. It then places the corners at the corners, and the middle sections are repeated or stretched as you specify.

**Note:** For border-image to work, the element also needs the border property set!

**Note**: The border-image property is actually a shorthand property for the border-image-source, border-image-slice, border-image-w0069dth, border-image-outset and border-image-repeat properties.

## CSS3 Background Size: The CSS3 background-size property allows you to specify the size of background images. Before CSS3, the size of a background image was the actual size of the image. CSS3 allows us to re-use background images in different contexts. The size can be specified in lengths, percentages, or by using one of the two keywords: contain or cover.

The two other possible values for background-size are contain and cover.

The contain keyword scales the background image to be as large as possible (but both its width and its height must fit inside the content area). As such, depending on the proportions of the background image and the background positioning area, there may be some areas of the background which are not covered by the background image.

The cover keyword scales the background image so that the content area is completely covered by the background image (both its width and height are equal to or exceed the content area). As such, some parts of the background image may not be visible in the background positioning area.

The background-size property also accepts multiple values for background size (using a comma-separated list), when working with multiple backgrounds.

Now we want to have a background image on a website that covers the entire browser window at all times.The requirements are as follows:

* Fill the entire page with the image (no white space)
* Scale image as needed
* Center image on page
* Do not cause scrollbars

Eg: html { background: url(img\_flower.jpg) no-repeat center fixed;   
    background-size: cover;  
}

**CSS3 background-origin Property:** The CSS3 background-origin property specifies where the background image is positioned.The property takes three different values:

* border-box - the background image starts from the upper left corner of the border
* padding-box - (default) the background image starts from the upper left corner of the padding edge
* content-box - the background image starts from the upper left corner of the content

**CSS3 background-clip Property:** The CSS3 background-clip property specifies the painting area of the background.The property takes three different values:

* border-box - (default) the background is painted to the outside edge of the border
* padding-box - the background is painted to the outside edge of the padding
* content-box - the background is painted within the content box

**CSS3 Colors:** In addition, CSS3 also introduces:

* RGBA colors
* HSL colors
* HSLA colors
* opacity

RGBA color values are an extension of RGB color values with an alpha channel - which specifies the opacity for a color. An RGBA color value is specified with: rgba(red, green, blue, alpha). The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

HSL stands for Hue, Saturation and Lightness. An HSL color value is specified with: hsl(hue, saturation, lightness)

HSLA color values are an extension of HSL color values with an alpha channel - which specifies the opacity for a color. An HSLA color value is specified with: hsla(hue, saturation, lightness, alpha), where the alpha parameter defines the opacity. The alpha parameter is a number between 0.0 (fully transparent) and 1.0 (fully opaque).

The CSS3 opacity property sets the opacity for the whole element (both background color and text will be opaque/transparent).The opacity property value must be a number between 0.0 (fully transparent) and 1.0 (fully opaque).

**CSS3 gradients:** let you display smooth transitions between two or more specified colors. Earlier, you had to use images for these effects. However, by using CSS3 gradients you can reduce download time and bandwidth usage. In addition, elements with gradients look better when zoomed, because the gradient is generated by the browser.

CSS3 defines two types of gradients:

* Linear Gradients (goes down/up/left/right/diagonally)
* Radial Gradients (defined by their center)

## CSS3 Linear Gradients: To create a linear gradient you must define at least two-color stops. Color stops are the colors you want to render smooth transitions among. You can also set a starting point and a direction (or an angle) along with the gradient effect.

Syntax: background: linear-gradient (*direction*, *color-stop1*, *color-stop2, ...*); Top-to-bottom is default.

background: linear-gradient (to right, red, yellow); /\* Standard syntax \*/

**Linear Gradient – Diagonal:** You can make a gradient diagonally by specifying both the horizontal and vertical starting positions. The following example shows a linear gradient that starts at top left (and goes to bottom right). It starts red, transitioning to yellow:

If you want more control over the direction of the gradient, you can define an angle, instead of the predefined directions (to bottom, to top, to right, to left, to bottom right, etc.). The angle is specified as an angle between a horizontal line and the gradient line.

## background: linear-gradient (*angle*, *color-stop1*, *color-stop2*); Eg: background: linear-gradient (-90deg, red, yellow);

Eg: with multiple color background: linear-gradient (red, yellow, green);

**Repeating a linear-gradient**: The repeating-linear-gradient () function is used to repeat linear gradients:

Eg: background: repeating-linear-gradient(red, yellow 10%, green 20%);

## CSS3 Radial Gradients: A radial gradient is defined by its center. To create a radial gradient, you must also define at least two-color stops.

## background: radial-gradient (*shape size*at*position, start-color, ..., last-color*);

By default, shape is ellipse, size is farthest-corner, and position is center. **Radial Gradient - Evenly Spaced Color Stops (this is default)**

The shape parameter defines the shape. It can take the value circle or ellipse. The default value is ellipse. Eg : background: radial-gradient (circle, red, yellow, green);

## Repeating a radial-gradient: The repeating-radial-gradient () function is used to repeat radial gradients: Eg: background: repeating-radial-gradient (red, yellow 10%, green 15%);

## CSS3 Shadow Effects: With CSS3 you can add shadow to text and to elements

## CSS3 Text Shadow: The CSS3 text-shadow property applies shadow to text. In its simplest use, you only specify the horizontal shadow (2px) and the vertical shadow (2px):

Adding color : eg h1 { text-shadow: 2px 2px red; }

Adding blur: eg : h1 { text-shadow: 2px 2px 5px red;}

## Multiple Shadows: To add more than one shadow to the text, you can add a comma-separated list of shadow

## CSS3 box-shadow Property: The CSS3 box-shadow property applies shadow to elements. In its simplest use, you only specify the horizontal shadow and the vertical shadow:

Eg: div {box-shadow: 10px 10px;}

Adding color: div {box-shadow: 10px 10px grey;}

Adding blur: div {box-shadow: 10px 10px 5px grey;}

You can also add shadows to the ::before and ::after pseudo-elements, to create an interesting effect:

CSS3 Text: CSS3 contains several new text features. In this chapter, you will learn about the following text properties:

* text-overflow
* word-wrap
* word-break

The CSS3 text-overflow property specifies how overflowed content that is not displayed should be signaled to the user. It can be clip (where it cut down the text content) or ellipsis (where it signaled the user with extra content by …)

Word-wrap: The CSS3 word-wrap property allows long words to be able to be broken and wrap onto the next line.

Word-break: The CSS3 word-break property specifies line breaking rules. It have keep-all value (where it will break the word at hyphen) and break-all value (where it will break the word at any character).

Text-justify: The text-justify property specifies the justification method to use when text-align is set to "justify". It Can be,

text-justify: auto|inter-word|inter-ideograph|inter-cluster|distribute|kashida|trim|initial|inherit;

text-align-last: It will align the last line of paragraphs, it value can be left and right.

**CSS3 Transforms:** CSS3 transforms allow you to translate, rotate, scale, and skew elements. A transformation is an effect that lets an element change shape, size and position.CSS3 supports 2D and 3D transformations.

**Responsive Web Design**: 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.

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:

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

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:

## Size Content to The Viewport: Users are used to scroll websites vertically on both desktop and mobile devices - but not horizontally! So, if the user is forced to scroll horizontally, or zoom out, to see the whole web page it results in a poor user experience.

Some additional rules to follow:

**1. Do NOT use large fixed width elements -**For example, if an image is displayed at a width wider than the viewport it can cause the viewport to scroll horizontally. Remember to adjust this content to fit within the width of the viewport.

**2. Do NOT let the content rely on a particular viewport width to render well** - Since screen dimensions and width in CSS pixels vary widely between devices, content should not rely on a particular viewport width to render well.

**3. Use CSS media queries to apply different styling for small and large screens** - Setting large absolute CSS widths for page elements will cause the element to be too wide for the viewport on a smaller device. Instead, consider using relative width values, such as width: 100%. Also, be careful of using large absolute positioning values. It may cause the element to fall outside the viewport on small devices.

## A responsive grid-view often has 12 columns, and has a total width of 100%, and will shrink and expand as you resize the browser window.

## 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.

## Eg : \* { box-sizing: border-box;}

**@MediaQueries**: 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 tablets, iPhone, and Androids.

## A media query consists of a media type and can contain one or more expressions, which resolve to either true or false.

## Syntax : @media not|only mediatype and (expressions) {  CSS-Code; }