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)
- child selector (>)
- adjacent sibling selector (+)
- general sibling selector (~)

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

The child selector selects all elements that are the immediate children of a specified element.

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"

The general sibling selector selects all elements that are siblings of a specified element.

Difference between div.class_name and div .class_name:

div.class_name – It means any div element with a class of class_name.
div .class_name – It means any element which is an ancestor of div and has a class of class_name.

To select only special types of <input> tags:

```
input[type="text"] {
        Property: value;
}

 input[type="password"] {
        property: value;
}  etc.
```

Colors in CSS are most often specified by:

- a valid color name - like "red"
- an RGB value - like "rgb(255, 0, 0)"
- a HEX value - like "#ff0000"

HTML and CSS supports 140 standard color names.

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).

CSS background properties:

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

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.

The background image for a page can be set like this:

```
body {
    background-image: url("paper.gif");
}
```

Values for background-repeat:

```
background-repeat: repeat-x;
background-repeat: repeat-y;
background-repeat: no-repeat;
```

To specify that the background image should be fixed (will not scroll with the rest of the page), use the `background-attachment` property:

```
background-attachment: fixed;
```

added by CSS3:

- `background-size`
- `background-origin`
- `background-clip`

link - http://www.w3schools.com/css/css3_backgrounds.asp

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

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).

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, or thick.

The `border-color` property is used to set the color of the four borders.

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

The `margin` properties set the size of the white space outside the border.

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

Tip: Negative values are allowed.

You can set the margin property to `auto` to horizontally center the element within its container.

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 horizontal margins (left and right)! Only vertical margins (top and bottom)!

The CSS `padding` properties are used to generate space around content.
The padding clears an area around the content (inside the border) 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.

Note: 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!

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.

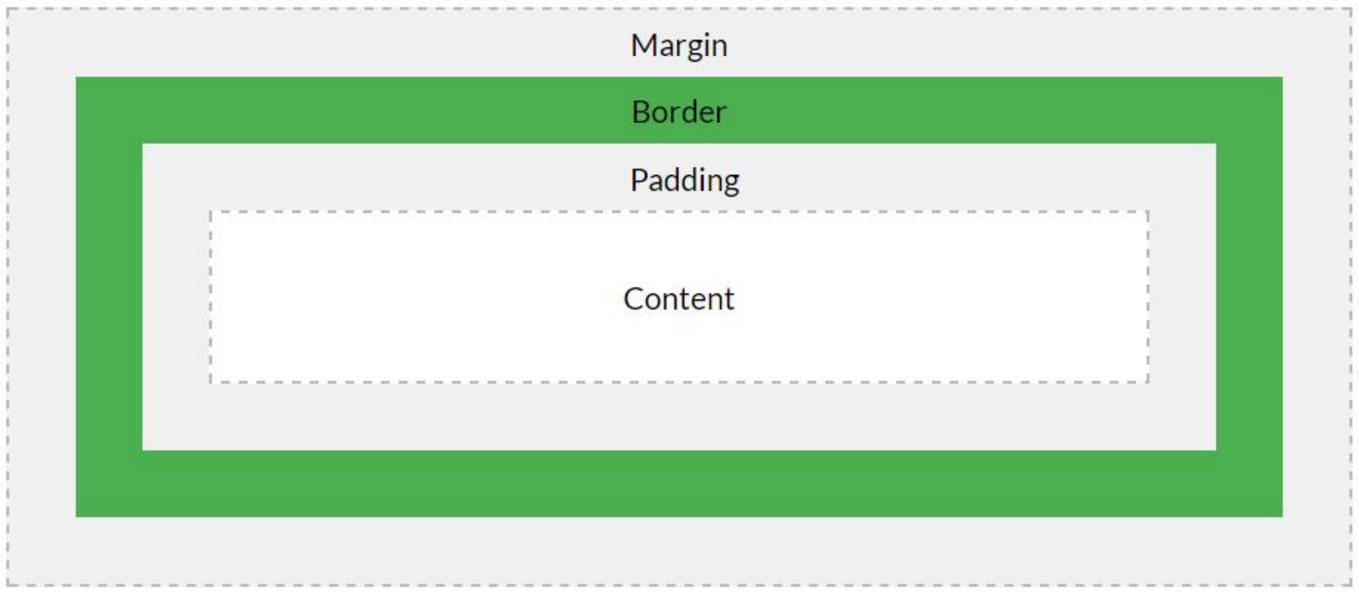
Using `max-width` instead, in this situation, will improve the browser's handling of small windows.

Note: The value of the `max-width` property overrides `width`.

Similar properties: max-height, min-width, min-height.

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
- The CSS `outline` properties specify the style, color, and width of an outline.
- An outline is a line that is drawn around elements (outside the borders) to make the element "stand out".
- However, the outline property is different from the border property - The outline is NOT a part of an element's dimensions; the element's total width and height is not affected by the width of the outline and it will overlap with any adjacent elements.
- All the syntax is same as border.

The `color` property is used to set the color of the text.

The `text-align` property is used to set the horizontal alignment of a text. Values are: left, right, center, justify. 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. Values are: overline, underline, line-through, none;

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. Values are: uppercase, lowercase, capitalize.

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. Values can be negative.

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. Values are: rtl, ltr.

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

The `text-shadow` property adds shadow to text. This property accepts a comma-separated list of shadows to be applied to the text.
Syntax: `text-shadow: h-shadow v-shadow blur-radius color|none|initial|inherit;`

The `white-space` property specifies how white-space inside an element is handled.
Syntax: `white-space: normal|nowrap|pre|pre-line|pre-wrap|initial|inherit;`

The `vertical-align` property sets the vertical alignment of an element.
Syntax: `vertical-align: baseline|length|sub|super|top|text-top|middle|bottom|text-bottom|initial|inherit;`
usage example link - http://www.w3schools.com/cssref/tryit.asp?filename=trycss_vertical-align

The CSS3 `text-overflow` property specifies how overflowed content that is not displayed should be signaled to the user. Values are: clip, ellipsis.
usage example link - http://www.w3schools.com/css/tryit.asp?filename=trycss3_text-overflow

The following table lists the new CSS3 text properties:

Property	Description
text-align-last	Specifies how to align the last line of a text
text-emphasis	A shorthand for setting text-emphasis-style and text-emphasis-color in one declaration
text-justify	Specifies how justified text should be aligned and spaced
text-overflow	Specifies how overflowed content that is not displayed should be signaled to the user
word-break	Specifies line breaking rules for non-CJK scripts
word-wrap	Allows long words to be able to be broken and wrap onto the next line

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:

```
p {
    font-family: "Times New Roman", Times, serif;
}
```

In CSS, there are two types of font family names:

- **generic family** - a group of font families with a similar look (like "Serif" or "Monospace")
- **font family** - a specific font family (like "Times New Roman" or "Arial")

Generic family	Font family	Description
Serif	Times New Roman Georgia	Serif fonts have small lines at the ends on some characters
Sans-serif	Arial Verdana	"Sans" means without - these fonts do not have the lines at the ends of characters
Monospace	Courier New Lucida Console	All monospace characters have the same width

The `font-style` property is mostly used to specify italic text.

The `font-size` property sets the size of the text. 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)

Tip: If you use pixels, you can still use the zoom tool to resize the entire page.

To allow users to resize the text (in the browser menu), many developers use em instead of pixels.

The em size unit is recommended by the W3C.

1em is equal to the current font size. The default text size in browsers is 16px. So, the default size of 1em is 16px.

The size can be calculated from pixels to em using this formula: *pixels*/16=*em*

To use google fonts, go to https://fonts.google.com/ , select a font, copy & paste the <link> tag in html and the font names in css.

The solution that works in all browsers, is to set a default font-size in percent for the <body> element:

```
body {
    font-size: 100%;
}
```

The `font-weight` property specifies the weight of a font. (for making text bold)

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 appears in a smaller font size than the original uppercase letters in the text.

How to add icons:

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

Add the name of the specified icon class to any inline HTML element (like `<i>` or `<span>`).

All the icons in the icon libraries below, are scalable vectors that can be customized with CSS (size, color, shadow, etc.)

To use the Font Awesome icons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/4.6.3/css/font-awesome.min.css">
```

```
<i class="fa fa-cloud"></i>
<i class="fa fa-heart"></i>
<i class="fa fa-car"></i>
<i class="fa fa-file"></i>
<i class="fa fa-bars"></i>
```

To use the Bootstrap glyphicons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
```

```
<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>
```

To use the Google icons, add the following line inside the `<head>` section of your HTML page:

```
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
```

```
<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>
```

The complete list of icons: http://www.w3schools.com/icons/icons_reference.asp

Example to use with hexadecimal values (have to include the ‘material-icons’ class for google API, ‘glyphicon’ class for bootstrap API, ‘fa’ class for font awesome API):

```
<i class="material-icons">&#xe540;</i>
<i class="fa">&#xf038;</i>
<i class="glyphicon">&#xe209;</i>
```

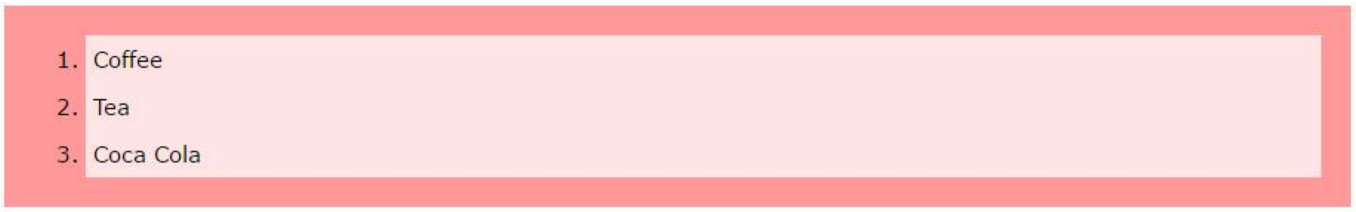
Links can be styled with any CSS property (e.g. `color`, `font-family`, `background`, etc.). In addition, links can be styled differently depending on what **state** they are in.

The four links states are:

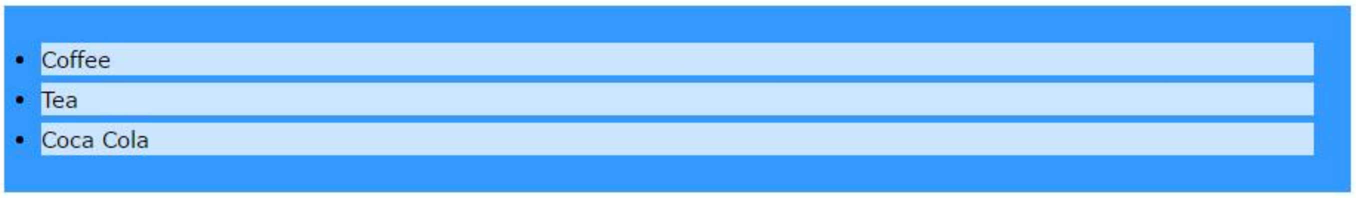
- `a:link` - a normal, unvisited link
- `a:visited` - a link the user has visited
- `a:hover` - a link when the user mouses over it
- `a:active` - a link the moment it is clicked

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



1. Coffee
2. Tea
3. Coca Cola



- Coffee
- Tea
- Coca Cola

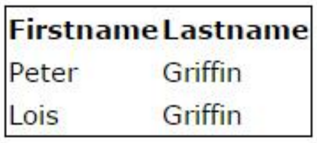
The `list-style-type` property specifies the type of list item marker.

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

```
ul {
    list-style-image: url('sqpurple.gif');
}
```

The `list-style-position` property specifies whether the list-item markers should appear inside or outside the content flow. Values are: inside, outside.

To specify table borders in CSS, use the `border` property. If you only want a border around the table, only specify the `border` property for <table>



Firstname	Lastname
Peter	Griffin
Lois	Griffin

The `border-collapse` property sets whether the table borders should be collapsed into a single border. Value is : collapse.

Width and height of a table are defined by the `width` and `height` properties.

The `text-align` property sets the horizontal alignment (like left, right, or center) of the content in <th> or <td>. By default, the content of <th> elements are center-aligned and the content of <td> elements are left-aligned.

The `vertical-align` property sets the vertical alignment (like top, bottom, or middle) of the content in <th> or <td>. By default, the vertical alignment of the content in a table is middle (for both <th> and <td> elements).

Use the `:hover` selector on <tr> to highlight table rows on mouse over.

CSS table properties:

Property	Description
border	Sets all the border properties in one declaration
border-collapse	Specifies whether or not table borders should be collapsed
border-spacing	Specifies the distance between the borders of adjacent cells
caption-side	Specifies the placement of a table caption
empty-cells	Specifies whether or not to display borders and background on empty cells in a table
table-layout	Sets the layout algorithm to be used for a table

The `display` property is the most important CSS property for controlling layout. 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`.

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).

An inline element does not start on a new line and only takes up as much width as necessary.

`display: none;` is commonly used with JavaScript to hide and show elements without deleting and recreating them.

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.

It has been possible for a long time to create a grid of boxes that fills the browser width and wraps nicely (when the browser is resized), by using the `float` property. However, the `inline-block` value of the `display` property makes this even easier. inline-block elements are like inline elements but they can have a width and a height.

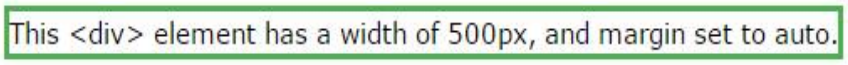
Hide an Element - display:none or visibility:hidden?

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.

As mentioned in the previous chapter; a block-level element always takes up the full width available (stretches out to the left and right as far as it can).

Setting the `width` of a block-level element will prevent it from stretching out to the edges of its container. Then, you can set the margins to auto, to horizontally center the element within its container. The element will take up the specified width, and the remaining space will be split equally between the two margins.


This <div> element has a width of 500px, and margin set to auto.

Note: The problem with the `<div>` above occurs when the browser window is smaller than the width of the element. The browser then adds a horizontal scrollbar to the page.

Using `max-width` instead, in this situation, will improve the browser's handling of small windows. This is important when making a site usable on small devices.

The `position` property specifies the type of positioning method used for an element. There are four different position values:

- `static`
- `relative`
- `fixed`
- `absolute`

Elements are then positioned using the top, bottom, left, and right properties. However, these properties will not work unless the `position` property is set first. They also work differently depending on the position value.

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.

An element with `position: relative;` is positioned relative to its normal position. Setting the top, right, bottom, and left properties of a relatively-positioned element will cause it to be adjusted away from its normal position. Other content will not be adjusted to fit into any gap left by the element.

An element with `position: fixed;` is positioned relative to the viewport, which means it always stays in the same place even if the page is scrolled. The top, right, bottom, and left properties are used to position the element. A fixed element does not leave a gap in the page where it would normally have been located.

Note: Absolute positioned elements are removed from the normal flow, and can overlap elements.

An element with `position: absolute;` is positioned relative to the nearest positioned ancestor (instead of positioned relative to the viewport, like fixed). However; if an absolute positioned element has no positioned ancestors, it uses the document body, and moves along with page scrolling.

Note: A "positioned" element is one whose position is anything except `static`.

All CSS Positioning Properties

Property	Description
bottom	Sets the bottom margin edge for a positioned box
clip	Clips an absolutely positioned element
cursor	Specifies the type of cursor to be displayed
left	Sets the left margin edge for a positioned box
overflow	Specifies what happens if content overflows an element's box
overflow-x	Specifies what to do with the left/right edges of the content if it overflows the

	element's content area
overflow-y	Specifies what to do with the top/bottom edges of the content if it overflows the element's content area
position	Specifies the type of positioning for an element
right	Sets the right margin edge for a positioned box
top	Sets the top margin edge for a positioned box
z-index	Sets the stack order of an element

When elements are positioned, they can overlap other elements. The z-index property specifies the stack order of an element. An element with greater stack order is always in front of an element with a lower stack order.

Note: z-index only works on positioned elements (position:absolute, position:relative, or position:fixed).

The CSS `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 - Default. The overflow is not clipped. It renders outside the element's box
- hidden - The overflow is clipped, and the rest of the content will be invisible
- scroll - The overflow is clipped, but a scrollbar is added to see the rest of the content. Note that this will add a scrollbar both horizontally and vertically (even if you do not need it)
- auto - If overflow is clipped, a scrollbar should be added to see the rest of the content. 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.

The `overflow-x` and `overflow-y` properties specifies whether to change the overflow of content just horizontally or vertically (or both):

`overflow-x` specifies what to do with the left/right edges of the content.
`overflow-y` specifies what to do with the top/bottom edges of the content.

The `float` property specifies whether or not an element should float. The `clear` property is used to control the behavior of floating elements.

If an element is taller than the element containing it, and it is floated, it will overflow outside of its container. Then we can add `overflow: auto;` to the containing element to fix this problem:

Aligning things in CSS - http://www.w3schools.com/css/css_align.asp

A pseudo-class is used to define a special state of an element.

For example, it can be used to:

- Style an element when a user mouses over it
- Style visited and unvisited links differently
- Style an element when it gets focus

The syntax of pseudo-classes:

```
        selector:pseudo-class {
            property:value;
        }
```

All CSS Pseudo Classes

Selector	Example	Example description
:active	a:active	Selects the active link
:checked	input:checked	Selects every checked <input> element
:disabled	input:disabled	Selects every disabled <input> element
:empty	p:empty	Selects every <p> element that has no children
:enabled	input:enabled	Selects every enabled <input> element
:first-child	p:first-child	Selects every <p> elements that is the first child of its parent
:first-of-type	p:first-of-type	Selects every <p> element that is the first <p> element of its parent
:focus	input:focus	Selects the <input> element that has focus
:hover	a:hover	Selects links on mouse over
:in-range	input:in-range	Selects <input> elements with a value within a specified range
:invalid	input:invalid	Selects all <input> elements with an invalid value
:lang(*language*)	p:lang(it)	Selects every <p> element with a lang attribute value starting with "it"
:last-child	p:last-child	Selects every <p> elements that is the last child of its parent
:last-of-type	p:last-of-type	Selects every <p> element that is the last <p> element of its parent

:link	a:link	Selects all unvisited links
:not(selector)	:not(p)	Selects every element that is not a <p> element
:nth-child(n)	p:nth-child(2)	Selects every <p> element that is the second child of its parent
:nth-last-child(n)	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)	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)	p:nth-of-type(2)	Selects every <p> element that is the second <p> element of its parent
:only-of-type	p:only-of-type	Selects every <p> element that is the only <p> element of its parent
:only-child	p:only-child	Selects every <p> element that is the only child of its parent
:optional	input:optional	Selects <input> elements with no "required" attribute
:out-of-range	input:out-of-range	Selects <input> elements with a value outside a specified range
:read-only	input:read-only	Selects <input> elements with a "readonly" attribute specified
:read-write	input:read-write	Selects <input> elements with no "readonly" attribute
:required	input:required	Selects <input> elements with a "required" attribute specified
:root	root	Selects the document's root element
:target	#news:target	Selects the current active #news element (clicked on a URL containing that anchor name)
:valid	input:valid	Selects all <input> elements with a valid value
:visited	a:visited	Selects all visited links

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

The syntax of pseudo-elements:

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

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**.

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

The following properties apply to the `::first-line` pseudo-element:

- font properties
- color properties
- background properties
- word-spacing
- letter-spacing
- text-decoration
- vertical-align
- text-transform
- line-height
- clear

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

Note: The `::first-letter` pseudo-element can only be applied to block-level elements.

The following properties apply to the ::first-letter pseudo- element:

- font properties
- color properties
- background properties
- margin properties
- padding properties
- border properties
- text-decoration
- vertical-align (only if "float" is "none")
- text-transform
- line-height
- float
- clear

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

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

The `::selection` pseudo-element matches the portion of an element that is selected by a user.The following CSS properties can be applied to `::selection`: `color`, `background`, `cursor`, and `outline`.

All CSS Pseudo Elements

Selector	Example	Example description
::after	p::after	Insert something after the content of each <p> element
::before	p::before	Insert something before the content of each <p> element
::first-letter	p::first-letter	Selects the first letter of each <p> element
::first-line	p::first-line	Selects the first line of each <p> element
::selection	p::selection	Selects the portion of an element that is selected by a user

CSS3 transition allows you to change property values smoothly (from one value to another), over a given duration.

To create a transition effect, you must specify two things:

- the CSS property you want to add an effect to
- the duration of the effect