CSS Pseudo cheat sheet

Simple selectors

Selector	Syntax	Example
Element	element	`div { }`
Class	.class	`.alpha { }`
ID	#id	`#alpha { }`
Universal	*	`* { }`

Variations of simple selectors

Elements	Syntax	Example	Description
Two classes	.first-class.second-class	`.alpha.beta { }`	All elements with classes alpha and beta
Element and class	element.class	`p.alpha { }`	All alpha class elements inside <p>
Two elements	element, element	`p, div { }`	All <p> and elements
Two elements	element element	`p div { }`	All elements inside <p>

Descendant selectors/combinators

Selector	Syntax	Example	Description
Descendant	element element	`div p { }`	All <p> descendants of
Child	element>element	`div > p { }`	All <p> direct descendants of
Adjacent Sibling	element+element	`div + p { }`	<p> element directly after
General Sibling	element~element	`div ~ p { }`	All <p> element iterations after

Attribute selectors

Selector	Syntax	Example
[attribute]	`[href] { }`	Selects all elements with a href attribute
[attribute=value]	`[lang="fr"] { }`	Selects all elements with lang attribute that has a value of "fr"
[attribute~=value]	`[input~=hello] { }`	Elements with input attribute containing the whitespace separated substring "hello"
[attribute|=value]	`[lang|=en] { }`	Elements with lang attribute value equal to "en" or "en-"(en hyphen)
[attribute^=value]	`a[href^="https"] { }`	Every <a> element with href attribute value begins with "https"
[attribute$=value]	`a[href$=".docx"] { }`	Every <a> element with href attribute value ends with ".docx"
[attribute*=value]	`a[href*="meta"] { }`	Every <a> element with href attribute value has substring "meta"

Peudo classes

Pseudo-class	Example	Description of selection
:active	`a:active { }`	All active links
:checked	`input:checked { }`	All the checked <input> elements
:default	`input:default { }`	All default <input> elements
:disabled	`input:disabled { }`	All disabled <input> elements
:empty	`div:empty { }`	All the elements with no children
:enabled	`input:enabled { }`	All the enabled <input> elements
:first-child	`p:first-child { }`	All the <p> elements who are the first child of a parent element
:first-of-type	`p:first-of-type { }`	All the <p> element who are the first <p> element of a parent element
:focus	`input:focus { }`	Input element under focus
:fullscreen	`:fullscreen { }`	The element in full-screen mode
:hover	`p:hover { }`	Action effect on mouse hover
:invalid	`input:invalid { }`	Input elements with an invalid value

Pseudo-class	Example	Description of selection
:last-child	`p:last-child { }`	All the <p> elements who are the last child of a parent element
:last-of-type	`p:last-of-type { }`	All the <p> elements who are the last <p> element of a parent element
:link	`a:link { }`	All unvisited links
:not(*selector*)	`:not(div) { }`	All the elements that are not a element
:nth-child(*n*)	`div:nth-child(3) { }`	All the <p> elements that are the third child of a parent element
:nth-last-child(*n*)	`div:nth-last-child(3) { }`	All the elements which are the third child of a parent element, counting from last child element
:nth-last-of-type(*n*)	`p:nth-last-of-type(2) { }`	The second sibling from the last child of a parent element.
:nth-of-type(*n*)	`p:nth-of-type(2) { }`	The second sibling of a parent element.
:only-of-type	`p:only-of-type { }`	All the <p> elements which are only <p> elements inside its parent
:only-child	`p:only-child { }`	All the <p> elements which are only child of a parent element
:optional	`input:optional { }`	The input elements with no "required" attribute
:required	`input:required { }`	Selects input elements with the "required" attribute specified
:root	`:root { }`	The Root element of document
::selection	`::selection { }`	The portion of an element that is selected by a user
:valid	`input:valid { }`	All the input elements with a valid value
:visited	`a:visited { }`	Selects all visited links

Pseudo-element selectors

Syntax	Example	Description
::after	`p::after { }`	Inserts content after content of <p> element
::before	`p::before { }`	Inserts content before content of <p> element
::first-letter	`p::first-letter { }`	Selects first letter of every <p> element
::first-line	`p::first-line { }`	Selects first line of every <p> element
::placeholder	`input::placeholder { }`	Selects input elements with "placeholder" attribute specified
::marker	`::marker { }`	Selects markers in a list

Text effects cheat sheet

The effects developers use on text items on a web page are chosen mainly because of their styling and layout style. Interesting effects can be created by combining these with other CSS properties.

The visual representation of text content can be changed by four main properties: text-transform, font-style, font-weight and text-decoration.

Property	**Values**	**Description**
Text-transform	None, uppercase, lowercase, capitalize, full-width	Modify text properties
Font-style	Normal, italic, oblique	Font styling options such as italics
Font-weight	Normal, weight, lighter, bolder, 100-900	Other font styling options like change of emphasis such as making text bold
Text-decoration	None, underline, overline, line-through	Shorthand for auxiliary elements added to text using other properties such as text-decoration-line

The additional properties that help configure styling effects are below.

Text-align	**For horizontal alignment of text**
Text-align-last	Alignment for the last line when text set to justify
Text-combine-upright	Multiple characters into the space of a single character placed upright like in Mandarin
Text-decoration-color	Color configuration of the text-decoration
Text-decoration-line	Line type in text-decoration such as underline, overline and so on
Text-decoration-style	Styles added to lines under text such as wavy, dotted and so on
Text-decoration-thickness	Thickness of the decoration line
Text-emphasis	Shorthand for other properties such as color and style
Text-indent	The indentation of the first line
Text-justify	Specifies the justification method used when text-align is "justify"
Text-orientation	Orientation of text in a line such as sideways, upright and so on
Text-shadow	Adds shadow to text
Text-underline-position	Declare position of underline set using the text-decoration property

Other than these, there are some more properties that help modify the alignment and define the scope of text with their containers.

Property	**Values**	**Description**
Text-overflow	Clip, ellipsis	Determines overflow behavior of text with the container
Word-wrap	Normal, anywhere, break-word	Applies to inline elements, alias for overflow-wrap
Word-break	Normal, break-all, keep-all, break-word	Used for long words to decide if words should break or overflow
Writing-mode	Horizontal-tb, vertical-lr, vertical-rl	Can set the text direction vertical or horizontal

The properties mentioned are ones that can be used for giving effects to text.