CSS (Cascading Style Sheets) in one page

Template: put the style sheet into the document

```
<html lang="en">

 <head>

  <title>... replace with your document's title ...</title>

  <style type="text/css">
   /* CSS Document */
   ... replace with your css's content ...
  </style>

 </head>

 <body>
  ... replace with your document's content ...
 </body>

</html>
```

Template: link to an external style sheet

your_document.html:

```
<html lang="en">

 <head>

  <title>... replace with your document's title ...</title>

  <link rel="stylesheet" type="text/css" href="your_document.css" />
  <link rel="stylesheet" type="text/css" media="print, handheld" href="another_document.css" />

 </head>

 <body>
  ... replace with your document's content ...
 </body>

</html>
```

your_document.css:

```
... replace with your css's content ...
```

```
/* Comment */

@import url("fancyfonts.css") me

@media media_type {
 selector {
  property: values;
  property: values;
 }
}

/* @media media_type is optiona
```

Media types (media_type)

all	For all devices
braille	For braille tactile feedback devices
embossed	For paged braille printers
handheld	For handheld devices
print	For paged material
projection	For projected presentations
screen	For color computer screens
speech	For speech synthesizers
tty	For media using a fixed-pitch character grid
tv	For television-type devices

More and details about media types >>>

Selectors

*	All elements (universal selector)
div	(type selector)
div *	Elements within
div span	within (descendant selector)

Units

px	Pixels
em	1 em equal to font size of parent (same as 100%)
ex	Height of lower case "x"
%	Percentage
in	Inches. 1 inch is equal to 2.54 centimeters.
cm	Centimeters
mm	Millimeters. 1 millimeter is equal to 1/10nd of an centimeter
pt	Points. 1 pt is equal to 1/72nd of an inch.
pc	Picas. 1 pc is equal to 12 pt
#789abc	RGB HEX Notation
#abc	Equates to "#aabbcc"
rgb(0,100,255)	Value (0 to 255) of each of red, green and blue
rgb(0%, 50%, 100%)	Value (0% to 100%) of each of red, green and blue
ms	Milliseconds
s	Seconds
Hz	Hertz

Main pages:
W3C Cascading Style Sheets Hor

Documentation:
Cascading Style Sheets, level 2
Syntax and basic data types >>
Selectors >>
Assigning property values, Casc
Media types >>
Box model >>
Visual formatting model >>
Visual formatting model details
Visual effects >>
Generated content, automatic nu
Paged media >>
Colors and Backgrounds >>
Fonts >>
Text >>
Tables >>
User interface >>

div, span	and (grouping selectors)
div > span	with as parent (child selector)
div + span	preceded by ... (adjacent sibling selector)
*.*any_class* .*any_class*	Elements of class="*any_class*" (class selector)
div.*any_class*	(class selector)
*#*item_id* #*item_id*	Element with id="*item_id*"> (aid selector)
div#*item_id*	(id selector)
*[*any_attr*] [*any_attr*]	Elements with sets *any_attr* attribute (attribute selector)
div[*any_attr*]	<div *any_attr*="..."> (attribute selector)
div[*any_attr*="*value*"]	<div *any_attr*="*value*"> (attribute selector)
div[*any_attr*~="*value*"]	<div *any_attr*="*another value another*"> (attribute selector)
div[*any_attr*|="*value*"]	<div *any_attr*="*value another*"> (attribute selector)
div:first-child	First child of (pseudo-class)
div:first-line	First line of (pseudo-elemen)
div:first-letter	First letter of (pseudo-element)
div:before	Element before (pseudo-elemen)
div:after	Element after (pseudo-element)
a:link :link	Non-active, unvisited links without mouse over (link pseudo-class)
a:visited :visited	Visited links (link pseudo-class)
div:hover	vhen mouse over (dynamic pseudo-class)
div:active	Active (dynamic pseudo-class)
div:focus	vith focus (dynamic pseudo-class)
div:lang(*la*)	with language *la* (language pseudo-class)

* Use any element instead div (span, p, a, h1, etc.)

More and details about selectors >>>

kHz	Kilohertz
0	0 requires no unit

More and details about lengths >>>

Aural style sheets >>
Default style sheet for HTML 4.
Full property table >>

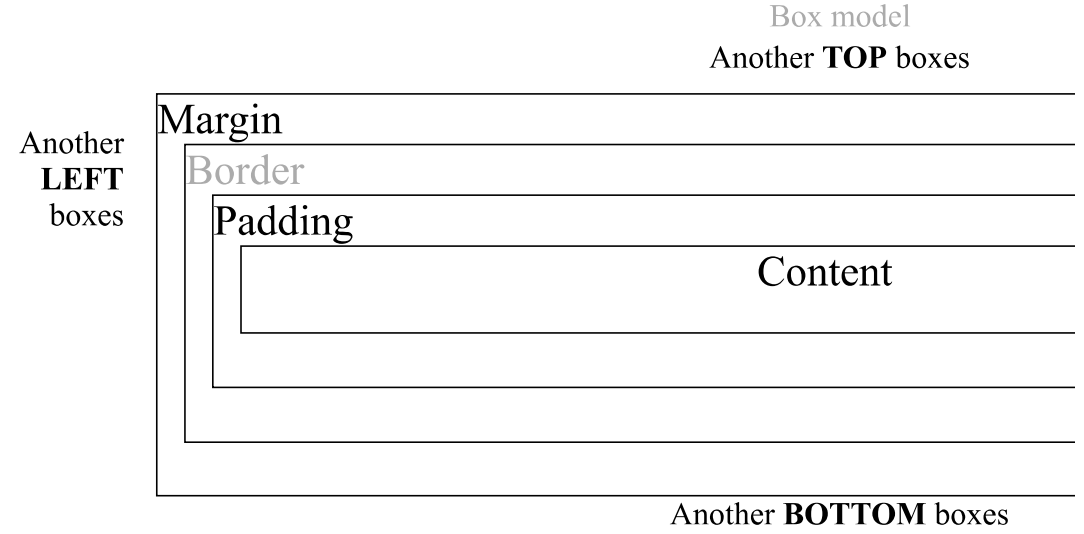
Tools

Validators:

- W3C MarkUp Validator. - Also known as the HTML validator, it helps check Web documents in formats like HTML and XHTML, SVG or MathML >>
- Checklink - Checks anchors (hyperlinks) in a HTML/XHTML document. Useful to find broken links, etc. >>
- CSS Validator - validates CSS stylesheets or documents using CSS stylesheets. >>
- RDF Validator >>
- Feed Validator. - it helps check newsfeeds in formats like ATOM, RSS of various flavors. >>
- P3P Validator - Checks whether a site is P3P enabled and controls protocol and syntax of Policy-Reference-File and Policy >>
- XML Schema Validator >>
- MUTAT - a human-centered testing tool (framework) >>

HTML/XHTML (HTML.SU)
XML - eXtensible Markup Langu
DTD - DocType Declaration
Other Manuals (MANUAL.SU)
Free archive of icons for downloa

Box model



Properties

Box

width	Specifies the content width of boxes generated by block-level and replaced elements	<length> | <percentage> | auto | inherit	p { width: 100px }
min-width	These two properties allow authors to constrain content widths to a certain range	<length> | <percentage> | inherit	h1 { min-width: 10px }
max-width		<length> | <percentage> | none | inherit	div { max-width: 600px }
height	specifies the content height of boxes generated by block-level, inline-block and replaced elements	<length> | <percentage> | auto | inherit	p { height: 150px }
min-height	These two properties allow authors to constrain content widths to a certain range	<length> | <percentage> | inherit	h1 { min-height: 10px }
max-height		<length> | <percentage> | none | inherit	div { max-height: 600px }
margin-top **margin-bottom** **margin-right**	Margin properties specify the width of the margin area of a box	<margin-width> | inherit Negative values for margin properties are allowed, but there may be implementation-specific limits.	**stylesheet fragment:** ul { background: yellow; margin: 12px 12px 12px 12px;

margin-left		
margin		
padding-top	Padding properties specify the width of the padding area of a box. The 'padding' shorthand property sets the padding for all four sides while the other padding properties only set their respective side.	<padding-width> | inherit
padding-bottom		
padding-right		
padding-left		
padding		
border-top-width	The border properties specify the width, color, and style of the border area of a box. These properties apply to all elements. **Note.** Notably for HTML, user agents may render borders for certain user interface elements (e.g., buttons, menus, etc.) differently than for "ordinary" elements.	thin - a thin border. medium - a medium border. thick - a thick border. <length> - the border's thickness has an explicit value. Explicit border widths cannot be negative.
border-right-width		
border-bottom-width		
border-left-width		
border-width		
border-top-color	The border color properties specify the color of a box's border	<color> | transparent | inherit
border-right-color		
border-bottom-color		
border-left-color		
border-color		
border-top-style	The border style properties specify the line style of a box's border (solid, double, dashed, etc.)	none - no border; the border width is zero. hidden - same as 'none', except in terms of border conflict resolution for table elements. dotted - the border is a series of dots. dashed - the border is a series of short line segments. solid - the border is a single line segment. double - the border is two solid lines groove - the border looks as though it were carved into the canvas ridge - the opposite of 'groove': the border looks as though it were coming out of the canvas inset - the border makes the box look as though it were embedded in the canvas outset - the opposite of 'inset': the border makes the box look as though it were coming out of the canvas
border-right-style		
border-bottom-style		
border-left-style		
border-style		
border-top	This is a shorthand property for setting the width, style, and color of the top, right, bottom, and left border of a box.	[<border-width> || <border-style> || <'border-top-color'>] | inherit
border-right		
border-bottom		
border-left		
border	Shorthand property for setting the same width, color, and style for all four borders of a box.	[<border-width> || <border-style> || <'border-top-color'>] | inherit

```
  padding: 3px 3px 3px 3px;
}

li {
  color: white;
  background: blue;
  margin: 12px 12px 12px 12px;
  padding: 12px 0px 12px 12px;
  list-style: none
}

#third {
  border-style: dashed;
  border-width: medium;
  border-color: lime;
}

#fourth {
  color: red;
  background: #ffa500;
  padding-top: 0.1em;
  padding-bottom: 2em;
  padding-left: 10em;
  padding-right: 1em;
  border-top-style: dotted;
  border-bottom-style: solid;
  border-left-style: double;
  border-right-style: groove;
  border-top-width: thin;
  border-bottom-width: thick;
  border-left-width: medium;
  border-right-width: medium;
  border-top-color: maroon;
  border-bottom-color: aqua;
  border-left-color: fuchsia;
  border-right-color: red;
}
```

html document fragment:

```
<ul>
  <li>first box</li>
  <li>second box</li>
  <li id="third">third box (with b
  <li><div id="fourth">fourth box
</div></li>
  <li>Attention! Some properties
some browsers!</li>
</ul>
```

Show boxes (Controlling box generation)

display	The values of this property have the different meanings	inline | block | list-item | run-in | inline-block | table | inline-table | table-row-group | table-header-group | table-footer-group | table-row | table-column-group | table-column | table-cell | table-caption | none | inherit
		block - this value causes an element to generate a block box
		inline-block - this value causes an element to generate a block box, which itself is flowed as a single

CSS fragment: em { display: blc
HTML fragment:

```
<em>First block</em><em>Secc
```

CSS fragment: em { display: inl

		inline box, similar to a replaced element. The inside of an inline-block is formatted as a block box, and the element itself is formatted as an inline replaced element	**HTML fragment:** First element box Second box
		inline - this value causes an element to generate one or more inline boxes	**CSS fragment:** p { display: inlin **HTML fragment:** First Second
		list-item - this value causes an element (e.g., LI in HTML) to generate a principal block box and a list-item inline box. For information about lists and examples of list formatting, please consult the section on lists.	**CSS fragment:** span { display: li **HTML fragment:** First Secor
		none - this value causes an element to generate no boxes in the formatting structure (i.e., the element has no effect on layout). Descendant elements do not generate any boxes either; this behavior cannot be overridden by setting the 'display' property on the descendants	**CSS fragment:** h3 { display: nor **HTML fragment:** First (hidden) Second
		run-in - this value creates either block or inline boxes, depending on context. Properties apply to run-in boxes based on their final status (inline-level or block-level).	**CSS fragment:** h3 { display: run **HTML fragment:** A run-in heading. And a paragraph of text that f
		table, inline-table, table-row-group, table-column, table-column-group, table-header-group, table-footer-group, table-row, table-cell, and table-caption - these values cause an element to behave like a table element	

Visual superposition of boxes (Positioning schemes)

position	The values of this property have the different meanings	static | relative | absolute | fixed | inherit	**CSS fragment for all examples:** p { display: block } span { displa
		static - the box is a normal box, laid out according to the normal flow. The 'top', 'right', 'bottom', and 'left' properties do not apply.	**CSS fragment:** #outer { position #inner { position: static; color: bl color: #FFFF99 } **HTML fragment:** Beginning of body contents. · class="outer">Start of outer cont class="inner"> Inner contents.</s contents.End of body con
		relative - the box's position is calculated according to the normal flow (this is called the position in normal flow). Then the box is offset relative to its normal position. When a box B is relatively positioned, the position of the following box is calculated as though B were not offset. The effect of 'position:relative' on table-row-group, table-header-group, table-footer-group, table-row, table-column-group, table-column, table-cell, and table-caption elements is undefined.	**CSS fragment:** #outer { position -12px; color: red } #inner { position: relative; top: 12 background-color: #FFFF99 } **HTML fragment:** Beginning of body contents. · class="outer">Start of outer cont class="inner"> Inner contents.</s contents.End of body con
		absolute - the box's position (and possibly size) is specified with the 'top', 'right', 'bottom', and 'left' properties. These properties specify offsets with respect to the box's containing block. Absolutely positioned boxes are taken out of the normal flow. This means they have no impact on the layout of later siblings. Also, though absolutely positioned boxes have margins, they do not collapse with any other margins.	Example №1 Example №2 Example №3 Example №4 (Implement change
		fixed - the box's position is calculated according to the 'absolute' model, but in addition, the box is fixed with respect to some reference. As with the 'absolute' model, the box's margins do not collapse with any other margins. In the case of handheld, projection, screen, tty, and tv media types, the box is	Example

		fixed with respect to the viewport and doesn't move when scrolled. In the case of the print media type, the box is rendered on every page, and is fixed with respect to the page box, even if the page is seen through a viewport (in the case of a print-preview, for example). For other media types, the presentation is undefined. Authors may wish to specify 'fixed' in a media-dependent way. For instance, an author may want a box to remain at the top of the viewport on the screen, but not at the top of each printed page.	
top	Specifies how far an absolutely positioned box's top margin edge is offset below the top edge of the box's containing block	<length> | <percentage> | auto | inherit	div.a8 { position: relative; directi
right	Specifies how far a box's right margin edge is offset to the left of the right edge of the box's containing block	<length> | <percentage> | auto | inherit	div.a8 { position: relative; directi
bottom	Specifies how far a box's bottom margin edge is offset above the bottom of the box's containing block	<length> | <percentage> | auto | inherit	div.a8 { position: relative; directi
left	Specifies how far a box's left margin edge is offset to the right of the left edge of the box's containing block	<length> | <percentage> | auto | inherit	div.a8 { position: relative; directi
float	This property specifies whether a box should float to the left, right, or not at all. It may be set for any element, but only applies to elements that generate boxes that are not absolutely positioned.	left | right | none | inherit	
		none - the box is not floated.	**CSS fragment:** #outer { color: re #inner { float: none; width: 130p background-color: #FFFF99 } #sibling { color: maroon } **HTML fragment:** <p>Beginning of body contents. class="outer">Start of outer cont class="inner"> Inner contents.</s class="sibling">Sibling contents. outer contents.End of boc
		left - the element generates a block box that is floated to the left. Content flows on the right side of the box, starting at the top (subject to the 'clear' property).	**CSS fragment:** #outer { color: re #inner { float: left; width: 130px; background-color: #FFFF99 } #sibling { color: maroon } **HTML fragment:** <p>Beginning of body contents. class="outer">Start of outer cont class="inner"> Inner contents.</s class="sibling">Sibling contents. outer contents.End of boc
		right - similar to 'left', except the box is floated to the right, and content flows on the left side of the box, starting at the top.	**CSS fragment:** #outer { color: re #inner { float: right; width: 130p background-color: #FFFF99 } #sibling { color: maroon } **HTML fragment:** <p>Beginning of body contents. class="outer">Start of outer cont class="inner"> Inner contents.</s class="sibling">Sibling contents. outer contents.End of boc
clear	Indicates which sides of an element's box(es) may not be adjacent to an earlier floating box. The 'clear' property does not consider floats inside the element itself or in other block formatting contexts.	none | left | right | both | inherit	
		left - the clearance of the generated box is set to the amount necessary to place the top border edge below the bottom outer edge of any left-floating boxes that resulted from elements earlier in the source document.	p { clear: left }
		right - the clearance of the generated box is set to the amount necessary to place the top border edge below the bottom outer edge of any right-floating boxes that resulted from elements earlier in the source document.	img { clear: right }
		both - the clearance of the generated box is set to the amount necessary to place the top border edge below the bottom outer edge of any right-floating and left-floating boxes that resulted from elements earlier in the source document.	div { clear: both }

		none - no constraint on the box's position with respect to floats.	em { clear: none }
z-index	'z-index' property specifies the stack level of the box in the current stacking context and whether the box establishes a local stacking context.	auto | <integer> | inherit <integer> - this integer is the stack level of the generated box in the current stacking context. The box also establishes a local stacking context in which its stack level is '0'. auto - the stack level of the generated box in the current stacking context is the same as its parent's box. The box does not establish a new local stacking context.	Example

Visual effects

overflow	Specifies whether content of a block-level element is clipped when it overflows the element's box	visible | hidden | scroll | auto | inherit	**HTML fragment for all exampl** <blockquote> <p>I didn't like the play, but th <cite>- Groucho Marx</cite> </blockquote>
		visible - indicates that content is not clipped, i.e., it may be rendered outside the block box	**CSS fragment:** div { overflow: visible; width : 1(100px; border: thin solid red; } blockquote { width: 125px; heigh top: 50px; margin-left: 50px; bor black } cite { display: block; text-align: r
		hidden - indicates that the content is clipped and that no scrolling user interface should be provided to view the content outside the clipping region.	**CSS fragment:** div { overflow: hidden; width : 1(100px; border: thin solid red; } blockquote { width: 125px; heigh top: 50px; margin-left: 50px; bor black } cite { display: block; text-align: r
		scroll - indicates that the content is clipped and that if the user agent uses a scrolling mechanism that is visible on the screen (such as a scroll bar or a panner), that mechanism should be displayed for a box whether or not any of its content is clipped. This avoids any problem with scrollbars appearing and disappearing in a dynamic environment. When this value is specified and the target medium is 'print', overflowing content may be printed.	**CSS fragment:** div { overflow: scroll; width : 10 border: thin solid red; } blockquote { width: 125px; heigh top: 50px; margin-left: 50px; bor black } cite { display: block; text-align: r
		auto - the behavior of the 'auto' value is user agent-dependent, but should cause a scrolling mechanism to be provided for overflowing boxes	**CSS fragment:** div { overflow: auto; width : 100 border: thin solid red; } blockquote { width: 125px; heigh top: 50px; margin-left: 50px; bor black } cite { display: block; text-align: r
clip	Applies only to absolutely positioned elements	<shape> | auto | inherit	p { clip: rect(5px, 40px, 45px, 5p

		In CSS 2.1, the only valid <shape> value is: rect(<top>, <right>, <bottom>, <left>) where <top> and <bottom> specify offsets from the top border edge of the box, and <right>, and <left> specify offsets from the left border edge of the box in left-to-right text and from the right border edge of the box in right-to-left text.	p { clip: rect(5px, 55px, 45px, 5p
visibility	The 'visibility' property specifies whether the boxes generated by an element are rendered. Invisible boxes still affect layout (set the 'display' property to 'none' to suppress box generation altogether).	visible | hidden | collapse | inherit visible - the generated box is visible hidden - the generated box is invisible (fully transparent, nothing is drawn), but still affects layout. Furthermore, descendents of the element will be visible if they have 'visibility: visible'. collapse - please consult the section on dynamic row and column effects in tables. If used on elements other than rows, row groups, columns, or column groups, 'collapse' has the same meaning as 'hidden'.	Example

Colors and Background

color	The foreground color of an element's text content	maroon (#800000)	span { color: red }

background
background-color
	Sets the background color of an element	red (#ff0000; #f00; rgb(255,0,0); rgb(100%, 0%, 0%))
		orange (#ffa500)
		yellow (#ffff00; #ff0; rgb(255,255,0); rgb(100%, 100%, 0))
		olive (#808000)
		purple (#800080)
		fuchsia (#ff00ff; #f0f; rgb(255,0,255); rgb(100%, 0%, 100%))
		white (#ffffff; #fff; rgb(255,255,255); rgb(100%, 100%, 100%))
		lime (#00ff00; #0f0; rgb(0,255,0); rgb(0%, 100%, 0%))
		green (#008000)
		navy (#000080)
		blue (#0000ff; #00f; rgb(0,0,255); rgb(0%, 0%, 100%))
		aqua (#00ffff; #0ff; rgb(0,255,255); rgb(0%, 0%, 100%))
		teal (#008080)
		black (#000000; #000; rgb(0,0,0); rgb(0%, 0%, 0%))
		silver (#c0c0c0)
		gray (#808080)
			em { color: #ff0000 }
			h5 { color: #f00 }
			p { color: rgb(255,0,0) }
			div { color: rgb(100%, 0%, 0%) }
			body {color: navy }
			a:visited { color: #505050 }
			h1 { background-color: #F00 }
			body { color: black; background:
			body { color: white; background:
			css file:
			body {
			background: aqua;
			color: blue
			}
			p { color: red }
			span { color: #ffa500 }
			html file:
			Example: This is a co

background-image	Sets the background image of an element	<uri> | none | inherit	body { background-image: url("n
background-repeat	Specifies whether the image is repeated (tiled), and how	repeat | repeat-x | repeat-y | no-repeat | inherit	body { background: white url("pe
background-attachment	Specifies whether it is fixed with regard to the viewport ('fixed') or scrolls along with the containing block ('scroll')	scroll | fixed | inherit	body { background: red url("penc
background-position	Specified, this property specifies its initial position.	[[<percentage> | <length> | left | center | right] [<percentage> | <length> | top | center | bottom]?] | [[left | center | right] || [top | center | bottom]] | inherit	body { background: url("banner.j
background	The 'background' property is a shorthand property for setting the individual background properties	[<'background-color'> || <'background-image'> || <'background-repeat'> || <'background-attachment'> || <'background-position'>] | inherit	p { background: url("chess.png")

Fonts

font-family	Prioritized list of font family names and/or generic family names	[[<family-name>|<generic-family>] [,<family-name>|<generic-family>]*] | inherit	body { font-family: Gill, Helveti
		'serif' (e.g. Times)	em { font-family: serif }
		'sans-serif' (e.g. Helvetica)	p { font-family: sans-serif }
		'cursive' (e.g. Zapf-Chancery)	a { font-family: cursive }
		'fantasy' (e.g. Western)	**li { font-family: fantasy }**
		'monospace' (e.g. Courier)	ul { font-family: monospace }
font-style	Selects between normal (sometimes referred to as "roman" or "upright"), italic and oblique faces within a font family	normal	h5 { font-style: normal }
		italic	*ol { font-style: italic }*
		oblique	*h3 { font-style: oblique }*

font-variant	Selects between normal (sometimes referred to as "roman" or "upright"), italic and oblique faces within a font family	normal SMALL-CAPS	dl { font-variant: normal } DD { FONT-VARIANT: SMALL-CAPS }
font-weight	Selects between normal (sometimes referred to as "roman" or "upright"), italic and oblique faces within a font family	normal **bold** **bolder** lighter 100 200 300 400 (eq normal) 500 **600** **700 (eq bold)** **800** **900**	dl { font-weight: normal } **dt { font-weight: bold }** **dt { font-weight: bolder }** dt { font-weight: lighter } dt { font-weight: 100 } dt { font-weight: 200 } dt { font-weight: 300 } dt { font-weight: 400 } dt { font-weight: 500 } **dt { font-weight: 600 }** **dt { font-weight: 700 }** **dt { font-weight: 800 }** **dt { font-weight: 900 }**
font-size	Selects between normal (sometimes referred to as "roman" or "upright"), italic and oblique faces within a font family	<absolute-size> | <relative-size> | <length> | <percentage> | inherit <absolute-size>: [xx-small | x-small | small | medium | large | x-large | xx-large] <relative-size>: [larger | smaller] <length>: [px | pt | pc | ex | in | cm | mm] <percentage>: [em | %] * see units	h6 { font-size: xx-small } em { font-size: x-small } h5 {font-size: small } h4 { font-size: medium } h3 { font-size: large } h2 {font-size: x-large h1 {font-size: xx blockquote { font-size: larg p { font-size: 16px; } @media print { p { font-size: 12p em { font-size: 1.5em em { font-size: 150%
font	The 'font' property is, except as described below, a shorthand property for setting 'font-style', 'font-variant', 'font-weight', 'font-size', 'line-height' and 'font-family' at the same place in the style sheet. The syntax of this property is based on a traditional typographical shorthand notation to set multiple properties related to fonts.	[[<'font-style'>||<'font-variant'>||<'font-weight'>] ?<'font-size'>[/<'line-height'>]?<'font-family'>] | caption | icon | menu | message-box | small-caption | status-bar | inherit caption - The font used for captioned controls (e.g., buttons, drop-downs, etc.). icon - The font used to label icons. menu - The font used in menus (e.g., dropdown menus and menu lists). message-box - The font used in dialog boxes. small-caption - The font used for labeling small controls. status-bar - The font used in window status bars.	p { font: 12px/14px sans-serif } p { font: 80% sans-serif } p { font: x-large/110% ***p { font: bold italic large Pal*** **P { FONT: NORMAL SMALL-CAPS 120%/** span { font: caption } span { font: icon } span { font: menu } span { font: message-box } span { font: small-caption } span { font: status-bar}

Text

text-indent	Specifies the indentation of the first line of text in a block	<length> | <percentage> | inherit	p { text-indent: 16px } div { text-indent: 3em }
text-align	Describes how inline content of a block is aligned	left | right | center | justify | inherit	p { text-align: left } p { text-align: justify }
vertical-align	Affects the vertical positioning inside a line box of the boxes	baseline | sub | super | top | text-top | middle | bottom | text-bottom | <percentage> | <length> | inherit	div { }

	generated by an inline-level element		
		baseline - align the baseline of the box with the baseline of the parent box. If the box doesn't have a baseline, align the bottom margin edge with the parent's baseline.	div { vertical-align: baseline }
		middle - align the vertical midpoint of the box with the baseline of the parent box plus half the x-height of the parent.	div { vertical-align: middle }
		sub - lower the baseline of the box to the proper position for subscripts of the parent's box. (This value has no effect on the font size of the element's text.)	div { vertical-align: sub }
		super - raise the baseline of the box to the proper position for superscripts of the parent's box. (This value has no effect on the font size of the element's text.)	div { vertical-align: super }
		text-top - align the top of the box with the top of the parent's content area	div { vertical-align: text-top }
		text-bottom - align the bottom of the box with the bottom of the parent's content area	div { vertical-align: text-bottom }
		<percentage> - raise (positive value) or lower (negative value) the box by this distance (a percentage of the 'line-height' value). The value '0%' means the same as 'baseline'.	div { vertical-align: -20% }
		<length> - raise (positive value) or lower (negative value) the box by this distance. The value '0cm' means the same as 'baseline'.	div { vertical-align: 15px }
		top - align the top of the aligned subtree with the top of the line box.	div { vertical-align: top }
		bottom - align the bottom of the aligned subtree with the bottom of the line box.	div { vertical-align: bottom }
text-decoration	Describes decorations that are added to the text of an element using the element's color	none | [underline || overline || line-through || blink] | inherit none - produces no text decoration underline - each line of text is underlined overline - each line of text has a line above it ~~line-through - each line of text has a line through the middle~~ blink - text blinks (alternates between visible and invisible)	**stylesheet fragment:** blockquote { text-decoration: unc } em { display: block; } cite { color: fuchsia; } **html document fragment:** <blockquote> Help, help! I am under a hat! <cite>—GwieF</cite> </blockquote>
letter-spacing	Specifies spacing behavior between text characters	normal | <length> | inherit	blockquote { letter-spacing
word-spacing	Specifies spacing behavior between words	normal | <length> | inherit	h1 { word-spacing: 1em
line-height	specifies the minimal height of line boxes within the element	normal | <number> | <length> | <percentage> | inherit	h1 { line-height: normal } /* normal */
		<length> - the specified length is used in the calculation of the line box height. Negative values are illegal.	div { line-height: 1.2em } /* length */
		<number> - the used value of the property is this number multiplied by the element's font size. Negative values are illegal. The computed value is the same as the specified	div { line-height: 1.2 } /* number */
		<percentage> - the computed value of the property is this percentage multiplied by the element's computed font size. Negative values are illegal.	div { line-height: 55% } /* percentage */
text-transform	Controls capitalization effects of an element's text	Capitalize - Puts The First Character Of Each Word In Uppercase	P { Text-Transform: Capitalize }
		UPPERCASE - PUTS ALL CHARACTERS OF EACH WORD IN UPPERCASE	P { TEXT-TRANSFORM: UPPE
		lowercase - puts all characters of each word in lowercase	p { text-transform: lowercase }
		none - no capitalization effects	p { text-transform: none }
white-space	Directs user agents to collapse sequences of whitespace, and break lines as necessary to fill line boxes	normal | pre | nowrap | pre-wrap | pre-line | inherit normal - directs user agents to collapse sequences of whitespace, and break lines as necessary to fill line boxes.	p { white-space: normal }

		pre - prevents user agents from collapsing sequences of whitespace. Lines are only broken at newlines in the source, or at occurrences of "\A" in generated content	pre { white-space: pre }
		nowrap - collapses whitespace as for 'normal', but suppresses line breaks within text	td[nowrap] { white-space: nowra
		pre-wrap - revents user agents from collapsing sequences of whitespace. Lines are broken at newlines in the source, at occurrences of "\A" in generated content, and as necessary to fill line boxes	pre[wrap] { white-space: pre-wra
		pre-line - directs user agents to collapse sequences of whitespace. Lines are broken at newlines in the source, at occurrences of "\A" in generated content, and as necessary to fill line boxes	:before,:after { white-space: pre-l
direction	Specifies the base writing direction of blocks and the direction of embeddings and overrides (see 'unicode-bidi') for the Unicode bidirectional algorithm	ltr | rtl | inherit ltr - left-to-right direction. rtl - right-to-left direction.	**XML fragment:** <hebrew> <par>HEBREW1 HEBREW2 en <par>HEBREW6 <emph>HEBR </hebrew> <english> <par>english9 english10 english1 <par>english14 english15 englisl <par>english17 <he-quo>HEBRI </english>
unicode-bidi	Values for this property have the different meanings	normal | embed | bidi-override | inherit normal - the element does not open an additional level of embedding with respect to the bidirectional algorithm. For inline-level elements, implicit reordering works across element boundaries. embed - if the element is inline-level, this value opens an additional level of embedding with respect to the bidirectional algorithm. The direction of this embedding level is given by the 'direction' property. Inside the element, reordering is done implicitly. bidi-override - for inline-level elements this creates an override. For block-level, table-cell, table-caption, or inline-block elements this creates an override for inline-level descendents not within another block-level, table-cell, table-caption, or inline-block element. This means that inside the element, reordering is strictly in sequence according to the 'direction' property; the implicit part of the bidirectional algorithm is ignored.	**CSS fragment:** hebrew, he-quo {direction: rtl; un english {direction: ltr; unicode-bi **CSS fragment:** hebrew, english, par {display: blo emph {font-weight: bold}
		More and details about text >>> about visual formatting model ('width', 'height', 'line-height' and 'vertical-a	

Generated content

content	This property is used with the :before and :after pseudo-elements to generate content in a document.	normal | none | [<string> | <uri> | <counter> | attr(<identifier>) | open-quote | close-quote | no-open-quote | no-close-quote]+ | inherit none - the pseudo-element is not generated normal - computes to 'none' for the :before and :after pseudo-elements. <string> - text content (see the section on strings). <uri> - the value is a URI that designates an external resource (such as an image). If a user agent cannot display the resource it must ignore it. <counter> - counters may be specified with two different functions: 'counter()' or 'counters()'. The former has two forms: 'counter(name)' or 'counter(name, style)'. The generated text is the value of the innermost counter of the given name in scope at this pseudo-element; it is formatted in the indicated style ('decimal' by default). The latter function also has two forms: 'counters(name, string)' or 'counters(name, string, style)'. open-quote and close-quote - these values are replaced by the appropriate string from the 'quotes' property.	span:before { content: none } li:before { content: normal } **CSS fragment:** span:before { content: "Chapter: ' **HTML fragment:** this is a chapter **CSS fragment:** **HTML fragment:** **CSS fragment:** **HTML fragment:** **CSS fragment:** q:before { content: open-quote } close-quote }

			HTML fragment: <q>Quote me!</q>
		no-open-quote and no-close-quote - introduces no content, but increments (decrements) the level of nesting for quotes.	**CSS fragment:** q:before { content: no-open-quot content: no-close-quote }
			HTML fragment: <q>No quote me!</q>
		attr(X) - this function returns as a string the value of attribute X for the subject of the selector. The string is not parsed by the CSS processor. If the subject of the selector doesn't have an attribute X, an empty string is returned. The case-sensitivity of attribute names depends on the document language.	**CSS fragment:** **HTML fragment:**
quotes	This property specifies quotation marks for any number of embedded quotations.	[<string> <string>]+ | none | inherit	q:lang(en) { quotes: "" "" """ """ }
		none - the 'open-quote' and 'close-quote' values of the 'content' property produce no quotation marks.	**CSS fragment:** q:lang(no) { quotes: "«" "»" "" ""
		[<string> <string>]+ - values for the 'open-quote' and 'close-quote' values of the 'content' property are taken from this list of pairs of quotation marks (opening and closing). The first (leftmost) pair represents the outermost level of quotation, the second pair the first level of embedding, etc.	**HTML fragment:** <q>Trøndere gråter når <q>Vinsj deklamert.</q>
		Automatic counters and numbering	
counter-increment	Accepts one or more names of counters (identifiers), each one optionally followed by an integer. The integer indicates by how much the counter is incremented for every occurrence of the element. The default increment is 1. Zero and negative integers are allowed	[<identifier> <integer>?]+ | none | inherit	**CSS fragment:** h3:before { content: "Chapter " c counter-increment: chapter; } h3 { counter-reset: section; } h4:before { content: counter(chap counter(section) " "; counter-incr
counter-reset	contains a list of one or more names of counters, each one optionally followed by an integer. The integer gives the value that the counter is set to on each occurrence of the element. The default is 0.	[<identifier> <integer>?]+ | none | inherit	**HTML fragment:** First chapter First section S section Second chapter First section S section
		Lists	
list-style-type	Specifies appearance of the list item marker if 'list-style-image' has the value 'none' or if the image pointed to by the URI cannot be displayed. The value 'none' specifies no marker, otherwise there are three types of marker: glyphs, numbering systems, and alphabetic systems.	disc | circle | square | decimal | decimal-leading-zero | lower-roman | upper-roman | lower-greek | lower-latin | upper-latin | armenian | georgian | lower-alpha | upper-alpha | none | inherit	**HTML fragment for all examle** First Second
		disc - rendering depends on the user agent.	**CSS fragment:** ol { list-style-type: disc}
		circle - rendering depends on the user agent.	**CSS fragment:** ol { list-style-type: circle }
		square - rendering depends on the user agent.	**CSS fragment:** ol { list-style-type: square }
		decimal - decimal numbers, beginning with 1.	**CSS fragment:** ol { list-style-type: decimal }

		decimal-leading-zero - decimal numbers padded by initial zeros (e.g., 01, 02, 03, ..., 98, 99).	**CSS fragment:** ol { list-style-type: decimal-leadi
		lower-roman - lowercase roman numerals (i, ii, iii, iv, v, etc.).	**CSS fragment:** ol { list-style-type: lower-roman
		upper-roman - uppercase roman numerals (I, II, III, IV, V, etc.).	**CSS fragment:** ol { list-style-type: upper-roman
		georgian - traditional Georgian numbering (an, ban, gan, ..., he, tan, in, in-an, ...).	**CSS fragment:** ol { list-style-type: georgian }
		armenian - traditional Armenian numbering	**CSS fragment:** ol { list-style-type: armenian }
		lower-latin or lower-alpha - lowercase ascii letters (a, b, c, ... z).	**CSS fragment:** ol { list-style-type: lower-latin }
		upper-latin or upper-alpha - uppercase ascii letters (A, B, C, ... Z).	**CSS fragment:** ol { list-style-type: upper-alpha }
		lower-greek - lowercase classical Greek alpha, beta, gamma, ... (α, β, γ, ...)	**CSS fragment:** ol { list-style-type: lower-greek }
list-style-image	Sets the image that will be used as the list item marker	<uri> | none | inherit	ul { list-style-image: url("http://www.iconsfree.org/ico design.thesign.ro_tkcPainter_ima
list-style-position	Specifies the position of the marker box in the principal block box	inside | outside | inherit outside - the marker box is outside the principal block box inside - the marker box is the first inline box in the principal block box, after which the element's content flows.	**CSS fragment:** ul { list-style-position: outside } ul.compact { list-style-position: i **HTML fragment:** - first list item list item <ul class="compact"> - first<b - second list item
list-style	Is a shorthand notation for setting the three properties 'list-style-type', 'list-style-image', and 'list-style-position' at the same place in the style sheet	[<'list-style-type'> || <'list-style-position'> || <'list-style-image'>] | inherit	ul > li > ul { list-style: circle outs

Tables

caption-side	Specifies the position of the caption box with respect to the table box.	top | bottom | inherit	caption { caption-side: bottom; w
table-layout	Controls the algorithm used to lay out the table cells, rows, and columns.	auto | fixed | inherit	table { table-layout: fixed; margi
border-collapse	Selects a table's border model.	collapse | separate | inherit	table { border: outset 10pt; borde
border-spacing	Selects a table's border model.	<length> <length>? | inherit	
empty-cells	Controls the rendering of borders and backgrounds around cells that have no visible content	show | hide | inherit	table { empty-cells: show }

Miscellaneous