

Cascading Style Sheets (CSS)

CSS rules – general format:

Selector – identifies tag to be affected; may be HTML tag, class or ID

Declaration/Definition – enclosed in curly brackets

Property – identifies attribute being defined; followed by colon

Value – followed by optional semi-colon to separate multiple declarations

!important – optional, gives declaration top priority

selector {property: value !important;}

h1 {text-align: center !important;}

multiple declarations – separated by semi-colons, e.g. h1 {color: red; font-weight: bold;}

multiple selectors – separated by commas, e.g.: h1, h2, h3 {font-weight: bold;}

Comments in style sheets - /* Your comments here */

Old browsers that don't support CSS may display style rules as body text – to avoid this, enclose embedded style sheets in HTML comments <!-- <style>...</style>-->

Types of style sheets

1. **External/Linked** – allows multiple pages to reference the same styles (lowest priority)

<link rel="stylesheet" href="path/filename.css" media="screen">

You can reference more than one external style sheet – the last one has higher priority

2. **@import** – alternative to <link>, contained within the embedded style sheet

<style><!--

@import url(path/filename.css);

--></style>

Must be before embedded styles

3. **Embedded** – controls styles on that page

<style><!--

...rules here...

--></style>

4. **Inline** – controls attributes for a specific element (top priority)

<tag style="property: value; property2: value2">

e.g. <h1 style="font-size: 18pt; color: #ff0000">

Types of selectors

1. **Classes** – allows you to apply a consistent rule to different elements

a. **Independent Classes** – style can be applied to any type of element

In style sheet: .classname {property: value;}

In HTML: <tag class="classname">

b. **Dependent Classes** – style applies only to a specific type of element

In style sheet: selector.classname {property: value;} – e.g. h2.special {color: red;}

In HTML: <tag class="classname"> – e.g. <h2 class="special">

- c. **Anchor Pseudo-Classes** – applies different attributes to anchors in different states
Must occur **in this order**:
 - a:link {property: value;} – before a link has been visited
 - a:visited {property: value;} – after a link has been visited (history must be enabled)
 - a:hover {property: value;} – when the mouse pointer is over the link
 - a:active {property: value;} – when the link is being clicked?
- d. Combining independent classes & pseudo-classes
a.big:link {property: value;}
- 2. **ID's** – similar to classes, but used only once per page; can be manipulated with JavaScript
 - a. **Independent ID's** – can be applied to any type of element
In style sheet: #idname {property: value;}
In HTML: <tag id="classname">
 - b. **Dependent ID's** – applies only to a specific type of element
In style sheet: selector#idname {property: value;} – e.g. h2#special {color: red;}
In HTML: <tag id="classname"> – e.g. <h2 id="special">
- 3. **Contextual** – applies rule when specified combination of tags occurs
In style sheet: ul li b.classname {property: value;}
In HTML: <b class="classname">
- 4. **Span's** – applies rule with a continuous line of text without causing line breaks
In HTML: ...
- 5. **Div's** – applies rule to a block of text – may include positioning, etc.
In HTML: <div class="classname">...</div>

Inheritance – in the case of nested elements, child elements generally inherit the styles of the parent, unless specified differently

e.g. <h1> has the same text properties as <body>

Note: some properties are not inherited, e.g. margins, width, borders

Precedence of Style Sheets – from highest priority to lowest:

1. Inline
2. Embedded
3. @import
4. External/Linked
5. Browser defaults

Precedence of selectors – from highest to lowest:

1. ID's – e.g. #id {property: value;}
2. Dependent Classes – e.g. h2.classname {property: value;}
3. Independent Classes – e.g. .classname {property: value;}
4. Contextual – e.g. ul li b {property: value;}

Font properties

- **font-family** – e.g. `p {font-family: verdana, tahoma, arial, "century gothic", sans-serif;}`
Note: names with spaces must be enclosed in quotes
Generic families:
 - serif – e.g. Georgia, Garamond, Times New Roman
 - sans-serif – e.g. Verdana, Tahoma, Arial
 - cursive – e.g. Bradley Hand ITC, Comic Sans MS, Edwardian Script ITC
 - monospace – e.g. Courier New
 - fantasy – e.g. Curlz MT, Wingdings
- **font-size** – e.g. `p {font-size: 14pt;}` – note: no space between value & unit
Absolute sizes: in, mm, cm, pt, pc (picas) – 72 pts/in; 12pts/pc; 6pc/in
xx-small, x-small, small, medium, large, x-large, xx-large
Relative sizes: em (m-length), ex (x-height), % (relative to parent element)
larger, smaller
Device-dependent sizes: px – size will vary depending on screen resolution?
- **font-style** – e.g. `p {font-style: italic;}`
Values: normal (default), italic, oblique (not recommended)
- **font-variant** – e.g. `p {font-variant: small-caps;}`; values: normal, small-caps
- **font-weight** – e.g. `p {font-weight: italic;}`
Values: normal, bold, bolder (stronger than bold), lighter
OR – numerical value; 100 (light) to 900 (heavy); increments of 100
- **font** – grouped properties
Note: font-size and font-family must be last two values
`p {font: small-caps 700 24pt "times new roman", arial, sans-serif; color: #ff0000;}`
- **color** – determines foreground (text) color
Note: W3C doesn't support color names; use hexadecimal codes instead

Text properties

- **text-transform** – e.g. `p {text-transform: uppercase;}`
Values: uppercase, lowercase, capitalize, none (default)
Note: older browsers may not recognize this; may be best to type it uppercase if that's what you want
Good for formatting dynamically generated text for easier reading
- **text-decoration** – e.g. `p {text-decoration: underline;}`
Values: blink (doesn't work in IE), ~~line-through~~, underline, none (default)
- **text-indent** – indents first line of a paragraph – e.g. `p {text-indent: value;}` (length or %)
- **word-spacing** – adjusts the space between words; positive=increase; negative=closer
e.g. `p {word-spacing: normal;}` value= length or "normal"
- **letter-spacing** (kerning) – adjusts the space between letters; positive=increase; neg.=closer
e.g. `p {letter-spacing: 0.25em;}` value= length; "normal"; 0 – prevents justification
- **line-height** (leading) – usually determined by largest element (character, image) on the line
values: normal (default); number (e.g. 2= double-spaced); length; percent
to include in Font property, use `"/`, e.g. `h2 {font: italic small-caps 24pt/32px arial;}`
- **text-align** – values: left (default), center, right, justify (not recommended)

Lists

- **list-style-type** – specifies bullet style
values: disc (•), circle (◦), square (▪), decimal (.), none (no bullet)
- **list-style-position** – specifies alignment of bulleted text: outside (hanging indent); inside
- **list-style-image** – specifies alternate image as bullet – GIF, JPEG, PNG
e.g. selector {list-style-image: url(path/filename);}
- **list-style** – can change an unordered list to ordered
values: upper-roman (I), lower-roman (i), upper-alpha (A), lower-alpha (a)
Grouped values, e.g. li {list-style: url(bullet.gif) square outside;}

Note: many of these properties are not supported by Netscape Navigator and IE 4

<dl> - definition list – indented but not bulleted

<dt> - definition term

<dd> - definition description

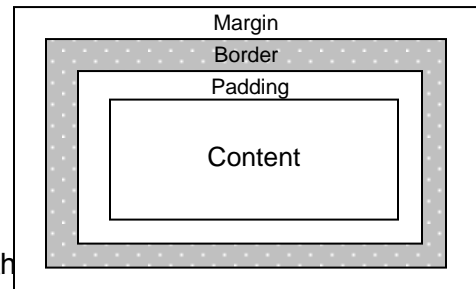
<blockquote>??

Background properties

- **background-color** – e.g. body {background-color: #FFCC00;}; value: name or hexadecimal
default value: none (transparent – parent element's background will show through)
- **background-image** – sets an image behind the element –
e.g. selector {background-image: url(path/filename);}
any unfilled space will be filled with the background color, if used
- **background-repeat** – defines tiling characteristics of background-image (required)
values: repeat (tiles), no-repeat (displayed once), repeat-x (repeats horizontally), repeat-y
- **background-position** – specifies placement of background-image (required)
values: length; percent; keywords (left, right, center for horiz.; top, bottom, center for vert.)
first value specifies horizontal; second specifies vertical
- **background-attachment** – does background-image (required) scroll or not
values: scroll, fixed
- **background** – grouped properties (must include background-image)
e.g. body {background: fixed no-repeat url(ball.gif) 50% 50%;}

Margin, Border & Padding

- **margin-top** –
- **margin-right** –
- **margin-bottom** –
- **margin-left** –
- **margin** – combined
 - One value – sets all 4
 - Two values – top & bottom; left & right
 - Three values – top; left & right; bottom
 - Four values – top, right, bottom, left
 - Values: length, percent (relative to parent element's width), auto (returns control to browser's discretion)
- **border-style** – values: none, solid, double, dashed, dotted, ridge, groove, inset, outset
 - Can set individual borders, e.g. {border-top-style: value;}
 - One, two, three or four values – same as Margin properties
- **border-width** – sets the thickness of the line
 - Values: length (0 makes it go away); keyword (thin, medium, thick)
 - Can set individual border, e.g. {border-top-width: value;}
- **border-color** – use hexadecimal values (e.g. {border-color: #FF0000;})
 - One, two, three or four values – same as Margin properties
- **border** – combines everything except border-color; must include border-style
 - Can set individual border, e.g. {border-top: width style color;}
- **height & width** – define size of border (element), e.g. p {width: 200px;}
 - Default – border stretches from left to right margin
 - Values: length, percent (relative to parent element's width), auto (Netscape isn't consistent)
- **Padding** – a
 - Values: length (0 makes it go away), percentage
 - Can set individual padding, e.g. {padding-top: value;}
 - One, two, three or four values – same as Margin properties



Float & Clear – allow control & flexibility of element placement

- **Float** – allows text to wrap around graphics, other blocks of text, etc., e.g. {float: left;}
 - Left or Right – specifies where element aligns, other elements wrap on the other side
 - None – defaults to parent element's alignment
 - NOTE: important to also declare Width for floated text elements; otherwise defaults to zero
- **Clear** – overrides Float property; e.g. {clear: left;}
 - Left or Right – moves element below other elements on left or right
 - Both – moves element below other elements on both sides
 - None – overrides other Clear settings; allows other elements to float to either side

Element placement

- **Browser windows** – two sizes:
 - **Browser window** – entire window, including any browser controls (e.g. scroll bars)
 - **Live browser window** – display area; always less than full window dimensions
- **Position** – specify position, shape, and stacking; e.g. {position: relative;}
 - **Static** – elements flow into document one after another; can't be explicitly positioned
 - **Relative** – in order of elements in code, but can be re-positioned
Relative to top-left corner of area where it would have been displayed if left static
 - **Absolute** – element is independent of all others; placed by x/y coordinates
Relative to origin (top-left) of Live window
 - **Fixed** – similar to Absolute, but doesn't move when window scrolls;
NOTE: doesn't work in most browsers
- **Top & Left** – specify position relative to parent or absolute grid
Values: length, percentage, auto (0 unless position: absolute; then browser-determined)
Positive values move down & right; negative values move up and left
- **Bottom & Right** – specify position relative to parent or absolute grid
Values: length, percentage, auto (0 unless position: absolute; then browser-determined)
Positive values move down & right; negative values move up and left
NOTE: be careful with "bottom" if element is longer than page height
- **Clip** – crops data or images too large for the container
Values: auto or rect(top right bottom left)
- **Overflow** – determines how to handle text that overflows container
Values: none, clip, scroll
- **Visibility** – shows or hides element on a page
Values: visible or hidden
- **z-index** – allows "stacking" elements – lower numbered elements appear below others
Values: positive (above), negative (below) or 0

Blocks

- **Div** – creates block of content; line breaks above and below.

Pseudo-elements

- **:first-letter** – e.g. p:first-letter {font-size: xx-large;} – creates a drop-cap
- **:first-line** – e.g. p:first-line {margin-left: 20px;} – creates a first-line indent

Printing properties

- **page-break-before** – e.g. <tag style="page-break-before: value";> only works as inline style
Values – always (forces page-break); auto; left (forces next page left-facing); right
- **page-break-after** –
Values – always (forces page-break); auto; left (forces next page left-facing); right