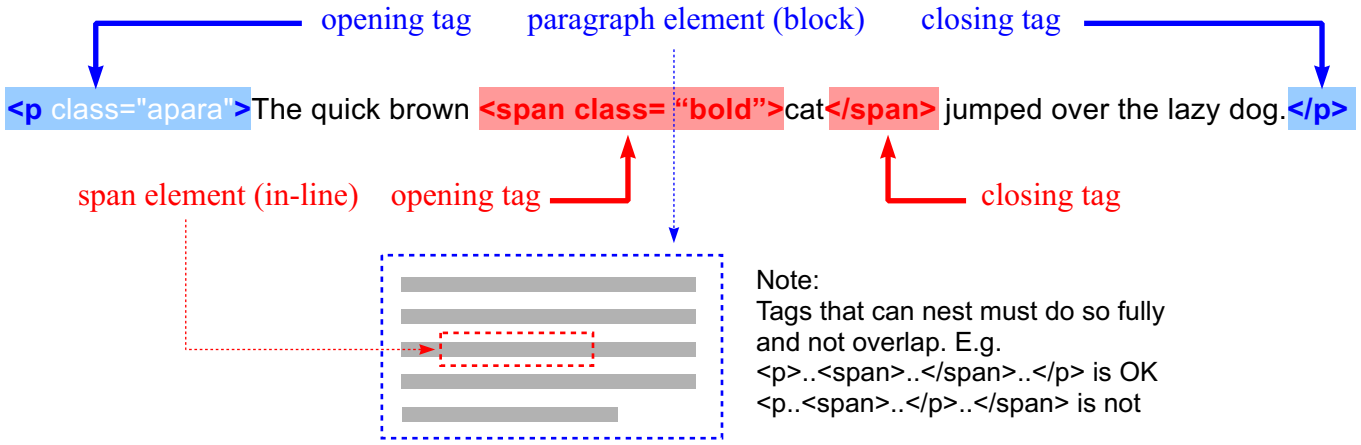
Computer code look-up

Xhtml

Hypertext mark-up language (HTML or XHTML) is the basic coding for a web page. The mark up instructions to the browser on how to display the readable text are contained within the text itself but identified by angle brackets. The instructions divide the text up into elements, each type having its own properties. Rather like word processor styles, some apply to complete blocks of text while others apply to text within a line while a third category define places for other items such as images and videos.

In addition to default characteristics, elements have settable attributes. Now, however, most of these have been replaced by instructions set by styles (cascading stylesheets, CSS) either directly through the style attribute of the element, a hidden instruction on the page or by a remotely linked style sheet.



Block elements are those that both start and end one or more sentences, be it a heading, paragraph, list or more, thus creating a tranche of the page.

Inline elements move with the text as it re-flows when browser window width is changed.

'Replaced' elements reserve an area on a page in which another file (such as an image) is displayed. Note that images are displayed on the page but are not embedded in it - they are provided separately and the location must be specified in the tag using the 'src' attribute and relative addressing.

Frame elements are for use when multiple pages are displayed in one window but CSS is preferred nowadays.

The page begins with a **doc type definition** which defines the standards which should be used by the browser in interpreting the code. Then there is a **head section** giving information about the document that is not dispayed in the window, using its own set of tags. The body element holds the visible content.

Core Attributes replace many of the original ones, now deprecated and set using CSS styles. These are:
ID can be used to uniquely identify any element within a page - useful handle for CSS or JavaScript
title often displayed as a tooltip when cursor comes over the element or while the element is loading
class used to associate an element with Cascading Style Sheet (CSS) rules defined externally.
style specifies Cascading Style Sheet (CSS) rules within the element

Deprecated Tags: applet , basefont , center , dir, embed, font, isindex, menu, plaintext, s, strike, u, tt, xmp
Deprecated Attributes: hspace, align, alink, bgcolor, border, height, link, nowrap, vlink, type, vspace

Special Character entities Not all text characters work in all environments, and have to be represented, so a table of special characters is included separately from this document. Click here

Head element tags	Function
<title></title>	the browser frame title, used by browsers to identify favourites
<link rel=stylesheet type="text/CSS" href="URL" />	links document to a style sheet, where URL is name & address of style sheet
<style type=text/CSS></style>	encloses and identifies styles applicable to the page itself. Protection by comment tag recommended
<script language="JavaScript"></script>	encloses and identifies a script which runs when the page opens. Protection by comment tag recommended
<meta name="keywords" content="key1, key2,etc." />	provides a search list for search engines
<meta name="description" content="description of site" />	provides a brief description for search engines
<meta HTTP-EQUIV="Refresh" content="5, URL" />	replaces page by another in the stated number of seconds
<meta HTTP-EQUIV="Expires" content="date & time" />	provides an expiry date for page

Replaced element tags	Comments and common attributes
<embed/> *not officially recognised but useful and widely supported*	attributes: src="compcoding.pdf" type="application/pdf" width="100%" height="600px" this is the code used to display this pdf document
<**img/>** *An image place-holder where src points to the image file*	attributes: src="image.gif" alt="descn for text-only browsers", + core attributes. (width, height, border, hspace, align usually set by CSS)
<**input** type="*see below*" value etc />	A box or button for user input
<**object**> </**object**>	embeds active x, java applet or other object, attributes according to type but including height and width
<**select**><**option**>first option <**option**>second option</**select**>	dropdown list for form. See inputs list input elements table.
also <audio/>, <canvas/>, <iframe>, <video/> are replaced elements not covered here	

Inline element tags	Comments and common attributes
<**a**></**a**>	hypertext link when used with attribute *href="file.htm#anchor"* Accesses a new web page (file.htm) or location within a page (#anchor) if included attribute *target="hierarchy"* specifies the frame or window for the new page
<**a** ></**a**>	anchor: the attribute *name="anchor"* defines a named point in document (anchor) for links to point to
<**br** />	line break: attribute *clear="all"* is used to prevent text riding up around a floating image. Single forward slash required only in XHTML
<**b**></**b**>, (or)	em**b**oldens text. If you really want bold, use with style attribute instead
<**i**></**i**> (sometimes <EM>)	**I**talic If you really want italic, use with style attribute instead
<**span**></**span**>	delineates a general purpose in-line selection of text, e.g. for use with style
<**sub**></**sub**>	**SUB**script.
<**sup**></**sup**>	**SUP**erscript.
<!-- comment -->	browser ignores anything inside

Block element tags	default characteristics	Blank sep line	Contains other blocks
<**body**></**body**>	The outermost container which delineates the page and contains everything visible. Every page has a **body** except a frameset.	No	Yes
<**div**></**div**>	**Division**, a multipurpose block for text with no default formatting. Can contain any element except body	No	Yes
<form></form>	Delineates a **form**, which can consist of text, tables, and form elements and be styled with CSS	Yes	Yes
<**h1**></**h1**>to <**h6**></**h6**>	**Heading**; Increasing text sizes are set by browser; h1 is usually has largest text. Can be overidden with CSS	Yes	No
<**p**></**p**>	**P**aragraph of text.	Yes	No
<blockquote></blockquote>	A para with both sides indented.	Yes	Yes
<**hr** />forward slash needed only in XHTML	horizontal line, settable attributes: size (height) in pixels, width in % or pixels, alignment and shading/no shading.	No	No
<**table**></**table**>	**table** attributes: width, border thickness cellspacing, cellpadding	No	Table rows only
<**td**></**td**>	TableCell attributes: width in % or pixels, valign (vertical alignment top middle or bottom), align (horiz alignment left, right, center)), rowspan or colspan (number of cells joined horizontally or vertically)	No	Yes
<**th**></**th**>	**T**able**H**ead holds a header row of cells whose contents are usually centred and boldened	No	Table cells only
<**tr**></**tr**>	**T**able**R**ow holds row of cells	No	Table cells only
<**ul**> list items see below</**ul**>	a bulleted list, attribute: type="zzz" where zzz=circle, disk or square	Yes	**ul** contains **li's** only
<**ol** type="n">list items see below</**ol**>	a numbered **list** where type can be 1, a, A or I	Yes	**ol** contains **li's** only
<**li**>list item</**li**>	Must be within ol, ul or li. Left indented. A number or bullet, is set by the enclosing ul or ol, outside the indent.	No	Yes
<**dl**>see below</**dl**>	a definition **list** - contains alternate term (dt) blocks and definition (dt) blocks	Yes	**dl's** contain **dt's** and **dd's**
<**dt**>word</**dt**> **dd**>definition</**dd**>	dd blocks are indented but dt blocks are not.	No	**dd's** yes

Frame elements	Better to use CSS to lay out your pages!
<**frame /**>	attributes: src="any.html" name="panename" marginwidth="2" marginheight="2" scrolling="0" sets frame name, scrolling requirement and starting file, nests inside frameset
<**frameset**> </frameset>	attributes: cols="50%,50%" rows="*,80" border="0" framespacing="0" frameborder="0" defines the size and arrangement of a set of frames and type of borders. 0 for no or 1 for yes Spacing in pixels
<**noframes**></noframes>	the text inside this element is displayed when the browser does not display frames

Input elements	Tag with attributes	notes
Text field	<input name="xx" type="text" maxlength="n", size="n" >	1,11
Radio button	<input name="xx" type="radio" >	2
Check box	<input name="xx" type="checkbox" value="ticked">	3,12
Reset button	<input name="xx" type="reset" >	4,9
Submit button	<input name="xx" type="submit" >	5,9
Ordinary button	<input name="xx" type="button" value="anything">	8,9,10
Drop down list	<select name="xx" size="2" ><OPTION value="optionvalue">list descn </select>	6,11
Text Area	<textarea name="xx" cols="n" rows="n" wrap="**" >	7,11

The **form input elements** are defined by their *type* attribute:

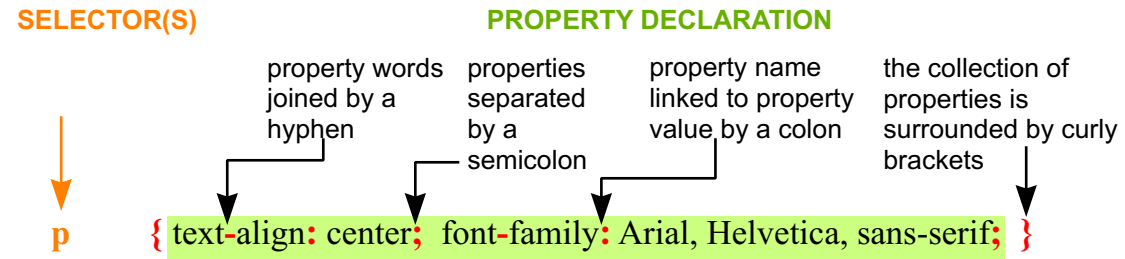
1. Maxlength refers to field length and size refers to max characters visible and size of the box.
2. To force a single choice give each radio element in the group the same name, add 'checked' attribute against the one you want as default selection if you want one.
3. Usually for multiple choice, add 'checked' against any default choices
4. Resets input values to default
5. Initiates action defined in form action
6. Place <option value=optionvalue>list description for each value in list and size=n for number of values visible, usually 1. If used, optionvalue is what goes into the name/value pair, whereas the visitor simply sees the list description.
7. Wrap options are off, *physical* to send as separate lines or *virtual* to send as one.
8. Use ordinary buttons to initiate JavaScript
9. To colour buttons, add a style attribute with color and background-color properties to the button opening tag. Change the border, etc.
10. The value of the button appears on the front of the button.
11. Add a style attribute with color and background-color properties to colour the text boxes.
12. Any value can be used here but ticked is readily understood.

Semantic elements are those where the name of the element describes the function, such as <list>, or <table>, and the W3C states that such elements should only be used for the purpose stated as any other use would be confusing for text-only readers, such as used for those without full sight. and imply no function and can be used to lay out pages and define appearance on the page together with CSS.

Cascading style sheets

Cascading styles (CSS) were brought out to separate the content of web pages (semantics) from the presentation (style). As a result, some HTML tags originally created to set styles have been deprecated and non-semantic elements and introduced to take CSS inline styles.

CSS acts on the basic HTML elements, to make them more readable, presentable, either directly from within the element tags (using the style attribute), or by being placed into the head of the page or on a separate, linked, style sheet. When not inside the element tag itself, a selector or identifier is used to identify which elements will be acted on, and properties define the styles themselves. The structure of a style is as follows:



Inside an HTML tag this becomes:

```
<p  style="text-align: center; font-family: Arial, Helvetica, sans-serif;  ">
```

Notice that each property has a name and a value, separated by a colon, and each property is separated from its neighbours by a semicolon.

SELECTORS

Selectors allow styles to be applied to multiple collections of HTML tags irrespective of where they appear on the page, according to predefined criteria. This makes updating easier.

Selector type	example	application
tag, or element	**h3**	all instances of the declared element. To apply to several elements separate them with a comma
class	**.greentext**	all elements with class="greentext" in the tag, classes are preceded by a dot To apply more than one class to an element, list them with a blank space btwn
ID	**#maintext**	an element with ID="maintext" in the tag - must be unique to the page, note the #
contextual	**li a**	apply to the last element only when it is (loosely) contained within the first element. Note there is a space but no comma between the selector tags
pseudo class (CSS2)	**a:hover**	applies to an element under certain conditions illustrated here by an element with the mouse over it
attribute	**[attribute]**	applies only when the tag contains the attribute or attribute/value in the square brackets
child	>	applies when second element is the direct child of the first
sibling	+	applies when the second element immediately follows the first

Examples of attaching styles

To stylesheet (see stylesheet linkage below)

```
ul.beerlist {
        line-height: 1.5;
        list-style-position: outside;
        list-style-image: url(https://www.tonero.me.uk/images/jug.gif);
        }
```

To document head

```
<style type="text/css">
<!--
ul.beerlist {
        line-height: 1.5;
        list-style-position: outside;
        list-style-image: url(https://www.tonero.me.uk/images/jug.gif);
        }
-->
</style>
```

To a class of element

Create the style in one of the two locations above, with the dot identifier, then apply the style to the elements as follows:

e.g.

To a unique element on page

Create the style in one of the two locations above, with the # identifier, then apply the style to the element as follows:

e.g.

To an element directly using style attribute

e.g. <ul style=”line-height: 1.5; list-style-position: outside; list-style-type: square; ”>

Note that the styles rolled onto one line and are enclosed by speech marks like any other attribute. If you are going to apply the style to more than one element it is more efficient to create a class.

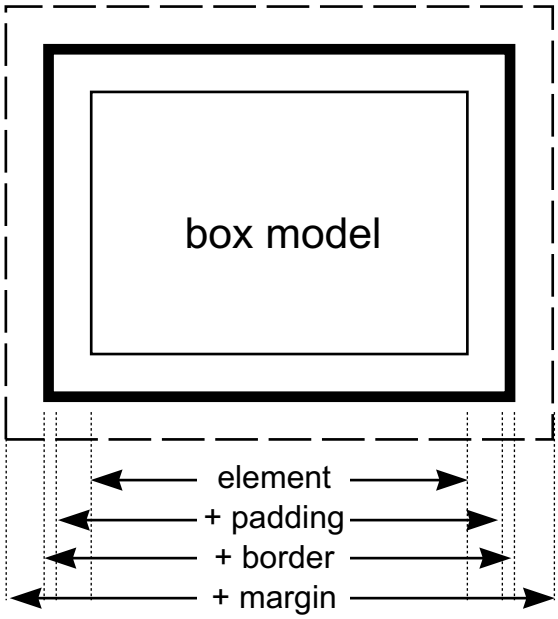
Two types of style sheet linkage

```
<link href="cssfiles/layoutstyles.css" rel="stylesheet" type="text/css" />

<style type="text/css" media="all">   @import "layoutstyles.css";   </style>
```

PROPERTIES

The box model



Each element is treated as a box from the point of view of applying style properties.

Padding (extension of the background outside the word wrap area), border and margins are not part of the element but are extras, so when an element size is specified, the width of padding, border and margin have to be added on.

In a vertical direction, where two margins abut, the shared margin will be made equal to the larger of the two constituent margins. In a horizontal direction (this is for inline elements) this is not the case.

Dimensions

- **absolute lengths:** in (inch), pc (pica, 6 per inch), pt (point, 72 per inch), mm, cm
- **relative lengths (rel to current element font-size):** em (for setting non-font properties)
- **relative lengths (rel to equivalent in parent):** em (for setting font properties), ex (x-height), %
- **relative lengths (other):** rem (rel html element font), vw, vh (rel width, height of viewport), vmax, vmin (rel max, min dimension of viewport) vw and vh *have only recently been supported*
- **Pixels:** px The pixel is not a unit of length but a quantity of the picture elements that make up the screen. CSS asks the browser to make allowances for the printer resolution when printing.

The em unit

Originally the width of an M, the em unit now refers to the font height, extending its application to fonts without an M. For the various block dimensions this is relative to the current text size, so as text size changes, the margins, padding etc. will change with it. The font em is related to its parent, so by setting everything to em units, the layout will scale automatically as the base font size is changed, as may be the case for those with sight impairment. vw is relatively new but may offer an alternative way of creating a flexible layout.

Applying properties to the edges of block elements

If applying to all four edges then you only need to apply once. e.g. *border: solid thin black*;
If applying to a single edge use *border-left, border-top* etc.
If applying to more than one edge give the values one after the other, clockwise from the top
e.g. *margin: 1em 3em 1em 2em;*
or for balancing pairs *margin: 1em 2em; gives top & bottom 1 em, left & right 2em*

Keywords

Many properties are expressed as keywords, which represent the quality or function to be applied.

Colour can be expressed as a keyword or in terms of hexadecimal - see page on computer colour.

Creating block elements from inline elements

Using the display property, block elements can become inline elements and vice- versa. This is particularly useful when designing animated menus. See animation below

Some common properties

Good references on the Internet: http://www.dhtmlgoodies.com/scripts/css-lookup/css-lookup.html and https://www.w3schools.com/css/default.asp and https://developer.mozilla.org/en-US/

name	example values and comments	applies to
background-attachment	scroll | fixed – sets background image to scroll with page or remain fixed	all elements
background-color	one of the 12 named colours| rgb(215, 0, 66) | rgb(0%, 50%, 3.5%) | #FF6653 | #746 – see color	all elements
background-image	url(imname.jpg) OR linear-gradient(direction, color1, color2, ...) if no direction e.g. *to bottom right* specified it defaults to to bottom	all elements
background-position	across and down from top left, act or % of visible area	all elements
background-repeat	repeat | repeat-x | repeat-y | no-repeat	all elements
border	set all 4 borders to same width, style, colour	all elements
border-collapse	collapse | separate – determines whether there is a gap between cells in a table. If all tables are same in document can set in body	table
border-color	one of the 12 named colours| rgb(215, 0, 66) | rgb(0%, 50%, 3.5%) | #FF6653 | #746 – see color For gradient see background-image	all elements
border-image	url(border.png) number [round | stretch] where number is the middle % of image to be repeated or stretched, e.g. 30. Use with border: solid transparent	all elements
border-radius	mm | cm | in | pt | pc (pica) | em | px | % (2 values each for ellipse) sets a rounded corner to all or to individual edges even if no border	all elements
border-spacing	only used for tables where border-collapse is set to separate. Use any length value; if two values takes first as horizontal	table elements
border-style	dotted | solid | dashed | double | groove | ridge | outset | inset | none (does all 4 borders the same)	all elements
border-width	dotted | solid | dashed | double | groove | ridge | outset | inset | none (does all 4 borders the same)	all elements
border-bottom	set bottom border width, style, colour (see border-width, border-style and border-color for values) not all need to be set	all elements
border-bottom-color	one of the 12 named colours| rgb(215, 0, 66) | rgb(0%, 50%, 3.5%) | #FF6653 | #746 – see color	all elements
border-bottom-left-radius	or -right-radius mm | cm | in | pt | pc (pica) | em | px | % sets a rounded corner bottom radius even if no border	all elements
border-bottom-style	dotted | solid | dashed | double | groove | ridge | outset | inset | none	all elements
border-bottom-width	thin | medium | thick | [value + mm | cm | in | pt | em | px | %]	all elements
border-left, border-right, border-top same applies as border-bottom		all elements
bottom	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | auto | inherit relative to height of containing clock	positioned elements

Properties continued

name	example values and comments	applies to
box-shadow	width, depth, blur, colour	all elements
caption-side	top | bottom | inherit determines whether a caption is displayed above or below a table	CAPTION
clear	none | left | right | both specifies on which sides a block element is not permitted to ride up alongside a floated element	block
clip	<shape> | auto | inherit clips an image within a preset shape e.g. clip: rect(5px, 310px, 250px, 10px} where the coordinates are given clockwise from top rel to top left	abs positioned elements
color	sets the colour of text aqua black blue fuchsia gray green lime maroon navy olive orange purple red silver teal white yellow | #FF6653 | #7F6 |rgb(215, 0, 66) | rgb(0%, 50%, 3.5%) |	any
cursor	auto | crosshair | pointer | move | e-resize | ne-resize etc. indicates direction of resize arrow | text | wait | help | url(images/mycursor.png) Determines how the cursor is represented in the element.	any
direction	ltr | rtl – governs the direction text appears along a line. Unlikely to be required as this will normally be done automatically	all elements
display	inline | block | list-item | none	all elements
empty-cells	show | hide – in the case of tables with separated cells this determines whether borders around empty cells will be displayed	table cells
float	left | right | none | inherit – the replacement for img align; allows an element to taken out of normal flow & placed to the left or right side	any
font	set all font properties(below) in one go, e.g. font: italic 1.3em arial;	all elements
font-family	specific e.g. Arial, Courier, Times (put preferred first, separate with commas and put speech marks round families with more than one word) | generic e.g serif, sans-serif, monospace, cursive, FANTASY	all elements
font-size	mm | cm | in | pt | pc (pica) | em | ex | % | xx-small | x-small | small | medium | large | x- large | xx-large | larger | smaller	all elements
font-stretch	ultra-condensed | extra-condensed | condensed | semi-condensed | normal | semi-expanded |expanded | extra-expanded | ultra-expanded |initial | inherit	all elements
font-style	normal | italic | oblique	all elements
font-variant	normal | small-caps	all elements
font-weight	normal | bold | bolder | lighter | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 (higher values are more bold)	all elements
left	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | auto | inherit	pos'd elems
letter-spacing	relative to height of containing clock	all elements
line-height	number (multiplies font size) | % (multiplies font size) | mm | cm | in | pt | pc (pica) | em | ex use with caution	all elements
list-style	allows a list to be defined in one go. See [type] [position] [image] below for values	display: list item
list-style-image	url("images/imagename.gif") can use absolute or relative addressing. If image is available, this overrides list-style-type	display: list item
list-style-position	inside | outside – sets the bullet or number to inside or outside the block	display: list item
list-style-type	circle | disc | square | decimal | upper-roman | lower-roman | none – sets the type of bullet or numbering	display: list item

Properties continued

name	example values and comments	applies to
margin	[value + mm | cm | in | pt | pc (pica) | em | px | % (of parent) | auto] sets the amount of space around a block. values are set clock-wise starting top e.g. *margin: 1em 2em;* gives top & bottom spacing 1 em and right and left spacing 2 em. Use of auto is complex.	all except certain table display types
note:	In a vertical direction, where two margins abut, the shared margin will be made equal to the larger of the two constituent margins.	
use margin-left, margin-right, margin-top, margin-bottom to set just one margin		
max-height	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | none | inherit	block, replaced
max-width	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | none | inherit	block, replaced
min-height	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | inherit	block, replaced
min-width	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | inherit	all elements
padding	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | auto values will be set clock-wise starting top e.g. *padding: 1em 2em;* for top & bottom padding 1 em and right and left padding 2 em.	all except some table
use padding-bottom, padding-left, padding-right, padding-top to set one padding		
page-break-after	auto | always | avoid | left | right – instructs printer to start a new page before this element	block
page-break-before	auto | always | avoid | left | right – instructs printer to start a new page before this element	block
page-break-inside	avoid | auto | inherit	block
position	static | relative | absolute | fixed | inherit	all elements
right	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | auto | inherit	positioned elements
text-align	left |center | right | justify	block
text-decoration	underline | overline | strike-through | blink | none | inherit	all
text-indent	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | inherit sets first line indent to value inserted	block, inline, td
text-shadow	width, depth, blur, colour	text
text-transform	capitalize | uppercase | lowercase | none – determines whether selected text is shown wholly in caps (uppercase) or init caps (capitalise) or not	all
top	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | auto | inherit	positioned elements
vertical-align	baseline | sub | top | text-top | middle | bottom | text-bottom | % (of line height) | mm | cm | in | pt | px sets selected text above or below the default baseline	inline & table cell
visibility	visible | hidden – when hidden the block does not display and the space is blank	all
white-space	normal | pre |nowrap | pre-wrap | pre-line | inherit	all
width	mm | cm | in | pt | pc (pica) | em | px | % of parent's width | auto | inherit e.g. not span	not non-replaced inline or table rows
word-spacing	mm | cm | in | pt | pc (pica) | em | px | % – increases (or decreases if -ve) the spacing by the amount set (em is preferred)	all elements
z-index	auto | <integer> | inherit	positioned elements

Animation

CSS can set properties to be dependent on the condition of browser items by using pseudo classes, creating a sort of animation: for menus and roll-overs in particular. These look at conditions of items such as links and set properties accordingly The commonest link conditions are: link, visited, hover and active. (link means unvisited link)

Used with the … tag they are represented by **a:link**, **a:visited**, **a:hover** and **a:active**

The order the CSS definition must appear in configuration in order to be effective is:
a:link a:visited a:hover a:active (**l v h a**)

Use an unordered list for navigation menus and remove the dots with text-decoration. Convert the link tag to a block to make the whole area clickable and colour responsive. To turn a vertical menu to a horizontal one float the list items. Use borders and padding to create the overall style.

e.g. #choosemenu a, #choosemenu **a:visited** { display: block; text-decoration: none; margin: 0 1px 0 0;
padding: 4px 8px; color: #006600; background: #bbccbb; }
#choosemenu **a:hover** { color: #060; background-color: #6f6; }

Later browsers allow the hover to be attached to other elements, useful for displaying tool tips.

Other pseudo classes work with form fields

An example on this site (mobindex.htm) uses a form field to expand a menu while hiding the form features from view. In practice this is frowned on as the form is not used as a form and may be confusing to text-only viewers.

#accordion label + input[type='radio']:checked + .content { display:block;} where a has a class called content

Some Pseudo Classes

selector	example	example modifies
:active	a:active	the active link
:checked	input:checked	checked <input> elements
:disabled	input:disabled	disabled <input> elements
:empty	p:empty	<p> elements that have no children
:enabled	input:enabled	enabled <input> elements
:first-child	p:first-child	<p> elements that are the first child of its parent
:first-of-type	p:first-of-type	<p> elements that are the first <p> element of its parent
:focus	input:focus	the <input> element that has focus
:hover	a:hover	elements on mouse over - links in this case
:in-range	input:in-range	<input> elements with a value within a specified range
:invalid	input:invalid	<input> elements with an invalid value
:lang(language)	p:lang(it)	<p> elements with a lang attribute value starting with "it"
:link	a:link	unvisited links
:not(selector)	:not(p)	elements except <p> element
:optional	input:optional	<input> elements with no "required" attribute
:out-of-range	input:out-of-range	<input> elements with a value outside a specified range
:read-only	input:read-only	<input> elements with a "readonly" attribute specified
:read-write	input:read-write	<input> elements with no "readonly" attribute
:required	input:required	<input> elements with a "required" attribute specified
:root	root	the document's root element
:valid	input:valid	<input> elements with a valid value
:visited	a:visited	visited links

Media queries

Use media queries to deliver a style sheet tailored specifically to desktops, laptops, tablets, mobile phones, printers, or screen readers (mediatype: print, screen, or speech). They will apply only when the media rule is valid.

Media Types
all - Default. Used for all media type devices
print - Used for printers
screen - Used for computer screens, tablets, smart-phones etc.
speech - Used for screenreaders that "reads" the page out loud

Linking via stylesheets:

```
<link rel="stylesheet" media="screen and (min-width: 900px)" href="widescreen.css">
<link rel="stylesheet" media="screen and (max-width: 600px)" href="smallscreen.css">
```

The **@media rule** is used in media queries to apply different styles for different media types/devices, eg to set the background blue or hide an element when screen size is 600px or less:.

```
@media only screen and (max-width: 600px) {
body {
background-color: lightblue;
}
}
```

```
@media screen and (max-width: 600px) {
div.example {
display: none;
}
}
```

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

Media features provide more specific details to media queries, by allowing to test for a specific feature of the user agent or display device. For example, you can apply styles to only those screens that are greater, or smaller, than a certain width.

Some Media Features
aspect-ratio The ratio between the width and the height of the viewport
color-index The number of colors the device can display
height, width The viewport height, width
hover Does the primary input mechanism allow the user to hover over elements?
max-aspect-ratio The maximum ratio between the width and the height of the display area
max-height The maximum height of the display area, such as a browser window
max-resolution The maximum resolution of the device, using dpi or dpcm
max-width The maximum width of the display area, such as a browser window
min-aspect-ratio The minimum ratio between the width and the height of the display area
min-height, -width The minimum height , width of the display area, such as a browser window
min-resolution The minimum resolution of the device, using dpi or dpcm
orientation The orientation of the viewport (landscape or portrait mode)
resolution The resolution of the output device, using dpi or dpcm
scripting Is scripting (e.g. JavaScript) available? (added in Media Queries Level 4)

Some principles of JavaScript

JavaScript is a text based code which can run alongside HTML. It can be used to create rollovers or other animation, validate forms, aid navigation, calculate results, display conditional text.

Not only can JavaScript modify the HTML it can also modify CSS styles applied to elements.

Each operation consists of an initiation, some logic and the modification or creation of one or more web page elements. Like HTML, JavaScript is in a state of transition. The original principles developed by the web browser companies are being superseded by a more powerful set based on the W3C Document Object model (W3C DOM) and which allows anything on a page to be monitored and manipulated.

Initiation	**Logic**	**Manipulation**
Parsing of document containing the code Detection of mouseover, mouseclick, onLoad, onFocus, onBlur, onChange etc. within an HTML element list below	fixed, conditional, mathematical, iterative, verification, get date etc.	Change an element's attributes or styles, Create, remove or move element(s), Change data in forms Alert, prevent or inform user

One cannot guarantee that JavaScript will be switched on.

The document object model(DOM)

The intent is that, starting from 'document', all elements are represented on the model as nodes, nested elements being branches from that node. Once you have identified an element you can travel along the branch to get at lower elements, (place a dot between each element in the chain in your code). Each element forms part of an array of elements of that type and will have a position within that array. Elements can have an HTML name and/or ID specified, in which case these can be used for identifying the element directly, although which of these is applicable depends on the browser.

Nodes have IDL definition, attributes/properties and methods, depending on node type. The HTMLDocument node is the top of the hierarchy and has the most attributes and methods assigned to it.

The document object has several useful methods for identifying elements on which you want to work:

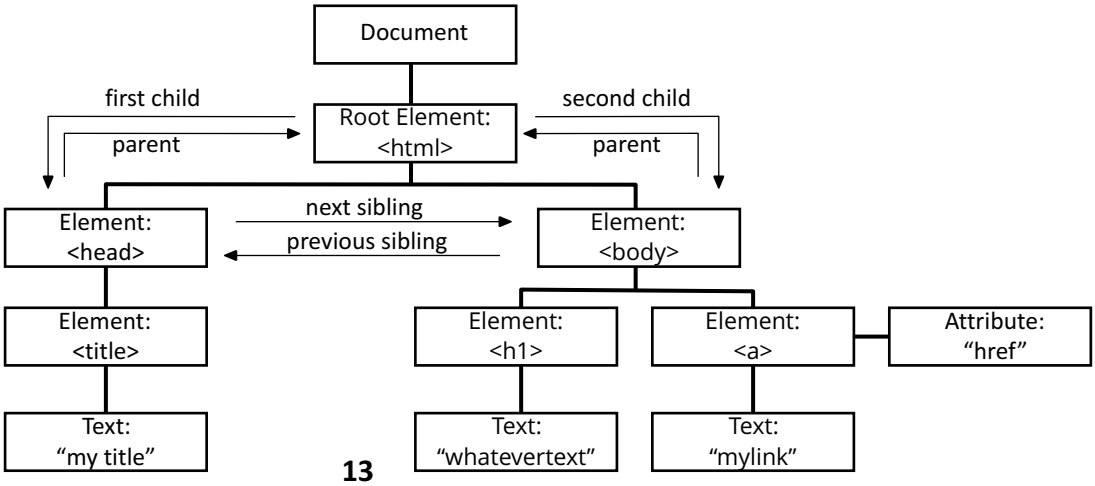
var variablename = document.getElementByID('*thegivenID*'); picks the element with the given ID from all elements in the document. From there you can walk the tree.

var variablename = document.getElementsByTagName('sometagname') [n]; picks the array holding all tags of type sometagname and extracts the specific one at array position n.

The W3C DOM

When the elements appear one after the other they are siblings and numbered accordingly.

Where nested they are in a parent child relationship and connected by a dot.



Syntax (bits in square brackets optional, not part of statement)

statement	each statement usually includes an assignment operator and ends in ;
function	declaration: function fnname([var name, var name]) {statements; [return var name]} use: functionname([value, value]) if a value is returned it is substituted for the function before continuing
conditional	if (condition is true) {statements;} [else { statements;}]
variable declaration	var anyname or var anyname = value or var anyname = new Array() var anyobject = new Object() or var anydate = new Date() etc.

Operators

assignment		**conditions**	
=	(x=y make x equal to y)	= =	equals
+=	(x+=y make x equal x + y)	!=	does not equal
-=	(x-=y make x equal x - y)	>	greater than
etc.		>=	greater than or equal
		&&	and
		||	or

Checking for existence

If you try to act on an object that doesn't exist, or use a method that is not supported by a particular browser then you will cause an error. You can use 'if' to check for existence before you call the item concerned:

if (object== null) {return} to check for an object or if (!document.getElementByID) {return} for a method

Styles

chosentag.style.propertyname = 'propertyvalue' where the property name and value are exactly as CSS except that dashes between property name words are replaced by intercaps, e.g. font-family becomes fontFamily.

Placing the code

The code can be placed on a separate sheet and linked, in the head of the document or in the body as a script element. The event handlers are usually placed as an attribute of the tag they are monitoring.

Linking a sheet:

<script type="text/javascript" src="/scripts/myscript.js"></script> Simply place the desired program on a plain sheet with a .js extension

Using a script tag:

<script type="text/javascript"> (optionally add language="JavaScriptn.n")
<!--
code in here…
//-->
</script>

Placing the event handler
This is added as an attribute to the tag being monitored.

Accessing elements via the DOM

getElementById("...") **By far the simplest way** is to set an element's ID then use getElementById to access this element. Then use .innerHTML to access the content if required.
e.g. *const someDivElement = document.getElementById("someDivElement")*

document.getElementsByClassName("...") This pulls out all elements with the specified class. Then you need to: either (1) pull the wanted element or (2) affect all elements.

getElementsByTagName("...") returns a list of all elements of a particular tag, e.g. p, div, li, ul,

querySelector("...") where ... can be tag, class etc. This will return a single element - if more occur in the document, it will return the first instance only.

querySelectorAll("...") Adding "All" to the previous method returns a list of all the element occurrences.

A complete list of event handlers. Code execution starts when the condition is met:

- onabort - playback interrupted;
- onafterprint - printing finished;
- onautocomplete - form autocomplete completed;
- onautocompleteerror - an error occurred while autocompleting the form;
- onbeforeprint - preparing for printing;
- onbeforeunload - the document is unloaded;
- onblur - loss of focus;
- oncancel - cancellation of the action;
- oncanplay - you can start playing the specified media file;
- oncanplaythrough - ditto without having to stop for buffering;
- onchange - value change;
- onclick - click on an element;
- onclose - closing something;
- oncontextmenu - opens the context menu;
- oncopy - copy performed;
- oncuechange - change the label in the track element;
- oncut - content was cut;
- ondblclick - double click on an element;
- ondrag - drag and drop an element;
- ondragend - element dragging completed;
- ondragenter - the element is dragged to a valid target area;
- ondragexit - exit drag-and-drop mode;
- ondragleave - the element leaves a valid target;
- ondragover - the element is dragged over a valid target point;
- ondragstart - start the drag-and-drop operation;
- ondrop - the dragged item is dropped;
- ondurationchange - change the length of the media;
- onemptied - the file suddenly became unavailable;
- onended - playback is over;
- onerror - an error occurred;
- onfocus - setting focus on an element;
- onhashchange - change the binding of a part of the address;
- oninput - start of data entry;

- oninvalid - the element is damaged;
- onkeydown - key pressed;
- onkeypress - key pressed and then released;
- onkeyup - key released;
- onload - the element is loaded;
- onloadeddata - file data loaded;
- onloadedmetadata - file metadata loaded;
- onloadstart - start loading an element;
- onmessage - message appears;
- onmousedown - mouse pressed;
- onmouseenter - the mouse is over the element;
- onmouseleave - the mouse pointer left the element;
- onmousemove - the mouse is moved over the element;
- onmouseout - the mouse pointer moves out of the element;
- onmouseover - the mouse pointer moves over the element;
- onmouseup - the mouse button is released over the element;
- onmousewheel (onwheel) - mouse wheel used;
- onoffline - the browser is running offline;
- ononline - the browser is running online;
- onpagehide - the user navigates from the page;
- onpageshow - the user goes to the page;
- onpaste - content was inserted;
- onpause - pause playback;
- onplay - start playback;
- onplaying - play the file;
- onpopstate - change the history of the window;
- onprogress - getting file metadata;
- onratechange - change the playback speed;
- onreset - data reset completed;
- onresize - resize the element;
- onscroll - scrolling the content of an element;
- onsearch - search performed;
- onseeked - search ended;
- onseeking - search is active;
- onselect - selection of some text or value;
- onshow - element display;
- onsort - performing sorting;
- onstalled - the browser cannot receive media for any reason;
- onstorage - updated web storage;
- onsubmit - confirmation of submitting form data;
- onsuspend - stop extracting metadata;
- ontimeupdate - change the position (time) of file playback, that is, rewind the file;
- ontoggle - the user opens or closes the details element;
- onunload - loading completed, after which the document was closed;
- onvolumechange - volume changed;
- onwaiting - waiting for playback to resume.

Some JavaScript snippets