Introduction To Web Development

- **A comprehensive learning guide**

HTML Tutorial

1. Introduction

[HTMLTutorialIntroduction]

Hypertext Markup Language (HTML) is one of the three main components of modern webpages, along with Cascading Style Sheets (CSS) and JavaScript. HTML indicates to the browser what elements should be included in the webpage (and in what order). CSS indicates how each element should be styled. JavaScript provides a means for webpage authors to manipulate these elements programmatically and in response to actions by the end user. Tutorials and reference material covering all three components are available here.

In these pages, we describe HTML further. Text used within HTML, CSS or JavaScript files is generally shown in `courier new` (i.e. a fixed space) font. The pages contain links to an extensive body of reference material explaining HTML, CSS and JavaScript in detail. We also provide a wide range of examples, which can help you understand better how HTML, CSS and JavaScript work. See below for further details on how to access these examples.

The concept of a markup language is explained further here. A document written in a markup language like HTML has parts that get rendered in the eventual output, but also parts that inform the rendering software how to interpret the remaining text. 'Rendering' here refers to the process of transforming the text document containing the HTML text into e.g. its visual representation on a screen.

The markup used by HTML includes tags, like `<p>…</p>`, to demarcate different HTML elements within the same webpage. In this case the `<p>` tag opens the relevant element and the `</p>` closes it. `<p>` elements are typically used to delimit paragraphs in HTML. HTML elements can be nested within other elements. Most elements can also be qualified by a range of attributes. For example, if we want to make the text within a `<p>` element appear red we can ascribe it a CSS style, along the lines of `<p style="color:red;">`.

Over time HTML has been refined. At the time of writing the latest version is HTML 5. Some aspects of earlier versions of HTML are no longer recognised in HTML 5 and some of these are noted where relevant.

Tutorial contents:

2. Getting started with HTML

[HTMLTutorialGettingStarted]

As explained in HTML and other markup languages, there are various 'dialects' of HTML. This means that some examples of HTML may be understood by some browsers but rejected by others. The following text, when put into a text editor and saved with a .htm file extension, will usually successfully render a web page that says "Hello World (using HTML)" if the file is viewed in Microsoft Edge. Note that HTML largely ignores page breaks; if you want to include a page break in the text shown to the user then you need to add a `<br>` element (or a `<br />` element if you are using XHTML, which is a modern variant of HTML that involves a cross between classic HTML and XML).

```
<html>
   <body>
       Hello World (using HTML)
   </body>
</html>
```

However, strictly speaking an HTML document is supposed to start with a document type declaration, along the lines of e.g. `<!DOCTYPE html>` and a header along the lines of e.g. `<head><title>`*Document title*`</title></head>`. So, a better way to create the page shown above is as follows. We've added a comment into the document, using HTML comment tags. Comments are not displayed by the browser but can help to document the HTML source text.

```
<!DOCTYPE html>
<html>
   <head>
      <title>My Document</title>
   </head>
   <!-- Only the text in the body will appear in the browser -->
   <body>
       Hello World (Using HTML)
   </body>
</html>
```

Often, the `<html>` element also includes a `lang` attribute, as this can be important for accessibility applications (such as screen readers) and for search engines. With the `lang` attribute, the first two letters specify the language. If the language comes in various dialects then two more letters specify the dialect, e.g.:

```
<html lang="en-US">
```

3. Carriage returns and thematic break lines

[HTMLTutorialLineBreaks]

HTML markup largely ignores carriage returns (i.e. line breaks) within marked up files. Instead, if you want to insert a carriage return you need to insert a `<br>` tag.

By 'largely ignores' we mean that browsers do not render carriage returns as line breaks as such (except if certain styles apply to the element within which they appear, see e.g. the default formatting of the <pre> element). However, carriage returns do affect how HTML handles spaces at the end of the line and at the start of the following line. Leading spaces (i.e. ones on a line before any non-space characters) are typically ignored, as are trailing spaces (i.e. ones on a line after the last non-space character). However, browsers typically insert a 'breaking space' at the end of each line, which often then shows up as a single space. Multiple spaces one after the other are interpreted as a single space. To include more than one space in such instances you need to include a 'non-breaking space' as a special character, see here.

For example the following markup:

```
Hello (followed by two carriage returns)<br><br />
   Hello
     again
and again
```

creates the following output:

Hello (followed by two carriage returns)

Hello again and again

Webpage authors typically put a lot of effort into creating visually appealing material. One way of breaking up text is to insert a thematic break line or horizontal rule, i.e. a `<hr>` tag, which places a line across the window like:

---

4. Commenting

[HTMLTutorialCommenting]

It can be helpful to include comments in HTML documents that are not displayed but help readers of the underlying markup text you to understand what the HTML is trying to do. Comments in HTML take the form `<!-- comment -->` and ignore line breaks within the opening and closing tags of the comment element.

For example, markup as follows:

```
<!-- material explaining how commenting in HTML
```

```
works-->
```

creates the following output (i.e. nothing):

5. Special characters

[HTMLTutorialSpecialCharacters]

The underlying markup of a webpage typically contains many more ampersand characters (i.e. `&`) than appear in the rendered output. This is because the `&` character is part of the way in which HTML marks up 'special' characters, i.e. ones that would otherwise be understood by HTML to relate to markup. In HTML, each special character is preceded by an ampersand, followed by the HTML markup name for that character followed by a semicolon. Perhaps the most common special characters are:

Special character	Meaning	HTML code
ampersand	&	`&`
space (technically a 'non-breaking' space)	(e.g. as in Hello again)	` ` (e.g. as in `Hello   again`)
less than sign	<	`<`
greater than sign	>	`>`
quotation mark	"	`&quot;`
apostrophe	'	`&apos;`

A fuller list of HTML special characters is available here.

6. Hyperlinks

[HTMLTutorialHyperlinks]

Many people associate web pages with hyperlinks, i.e. the ability to navigate from one page to another page. In HTML, hyperlinks (also called 'anchors') typically have the following sort of structure:

```
<a href="Pages/Aboutcns.pdf";>text</a>
```

The *text* is what the user sees, the value of `href` is where the link points to. Points to note include:

(a) The 'text' material seen by the user can contain HTML, so can include e.g. images and formatted text

(b) The `href` value used here, i.e. "`Pages/Aboutcns.pdf`" means that the link points to a webpage (or other resource) called "Aboutcns.pdf" in the directory "Pages" (strictly speaking a subdirectory of the directory in which the source webpage resides, unless it e.g. starts with `http://` or `https://` or unless the document's <base>element, if any, defines a different base address to be used by relative uniform resource locators, i.e. 'URLs').

The above link renders as:

text

Groups of hyperlinks can be included in a <nav> element. For example, markup as follows:

```
<nav>
<a href="Introduction.aspx">Introduction</a> |
<a href="IntroductionSoftware.aspx">Software</a>
</nav>
```

creates the following output, involving 2 individual hyperlinks:

Introduction | Software

7. HTML elements and their attributes

[HTMLTutorialHTMLElements]

The basic building blocks of HTML are elements (also called tags). A list of recognised elements is shown here. Some HTML elements, like the hyperlinks in HTMLTutorialHyperlinks, are by default differentiated from the text around them. The most general way of formatting text (capable of altering any of the default formatting of any visible HTML element) involves use of Cascading Style Sheets (CSS) or in-file or in-line equivalents, see cns CSS Tutorial. In-line CSS involves assigning a specific style attribute to a given element. Other sorts of attributes can also be assignedto different sorts of elements. A list of recognised attributes is shown here. The exact range of attributes valid for a specific element type does vary; see individual elements or attributes for further details.

Many other elements are also by default differentiated from the text around them or exist primarily to facilitate this sort of differentiation. Examples include:

HTML element	Normally used for e.g.	example
<address>	Contact information for the author or owner of a document	Mr. Smith, 1, George Street, Georgetown
<b>	Bold text	**1, George Street, Georgetown**
<blockquote>	Section that is quoted from another source	1, George Street, Georgetown
<cite>	Title of a work	Systemic Risk
<code>	A piece of computer code	`var x = 2.0;`
<del>	Indicates text deleted from a document	a~~bc~~
<dfn>	Defining instance of a term	*HTML*, a markup language
	Emphasised text (often used to italicise text, but ideally this should be done using CSS, as emphasis does not need to involve italics)	*HTML*, a markup language
<footer>	Footer for a document or section	HTML

<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	HTML headings	# Header 1 ## Header 2 ### Header 3 #### Header 4 ##### Header 5 ###### Header 6
<header>	Header for a document or section	HTML
<i>	A part of text in an alternate voice or mood	*HTML*, a markup language
<ins>	Indicates text added to a document	def
<kbd>	Keyboard input	`text representing keyboard input`
<mark>	Marked/highlighted text	text to highlight
<pre>	Preformatted text	`preformatted text (in fixed -width font that also preserves spaces and line breaks`
<q>	Short quotation	Mary had a little lamb
<s>	Text that is no longer correct	~~abc~~
<samp>	Sample output from a computer program	`output`
<small>	Smaller text	1, George Street, Georgetown
	Defines more important text, commonly used as another way of highlighting text or making it bold	**Mary had a little lamb**
<sub>	Subscripted text	A_1
<summary>	Heading for a <details> element	Heading
<sup>	Superscripted text	A^1
<time>	Date / time (N.B. isn't normally differentiated from standard text in most modern browsers)	10:00
<u>	Text that should be stylistically different from normal text, commonly used for underlining	abc
<var>	Variable	*Param1*
<wbr>	Posible line-break, the Word Break Opportunity tag specifies where in a text it would be ok to add a line-break	

Some HTML elements are no longer supported in HTML5. Although they often work in browsers you should ideally use CSS instead. These include:

HTML element	Normally used for e.g.	example
<big>	Big text	1, George Street, Georgetown
	Centred text	abc
<font> as in e.g. `<font color="green">`	Green text	green text
<strike>	Strikethrough text, not supported in HTML 5 (instead use <del> or <s>)	~~abc~~
<tt>	Teletype text	`abc`

Some HTML tags differentiate content, but primarily to assist the browser's understanding of that content, e.g.:

HTML element	Normally used for e.g.	example
<abbr> as in e.g. `<abbr title="Mister">`	Abbreviation or acronym	Mr.
<data> as in e.g. `<data value="1011">`	Links content with a machine-readable translation	Apple

Some HTML tags demarcate content so that the material can be segmented up more effectively, or assigned different formatting styles, e.g.:

HTML element	Normally used for e.g.	example
<article>	Article	**Title** Material
<aside>	Content aside from the page content	**Title** Material
<details>	Additional details that a user can view or hide	**Title** Material
	Section in a document	a single piece of content
<main>	Main content of a document	main text
<p>	Paragraph (by default has space added above and below the text)	a paragraph
<section>	Section in a document	a section
	Section in a document	a section (span)

A summary of the default styles applied to each HTML element is set out here.

CSS Tutorial

1. Introduction

[CSSTutorialIntroduction]

Cascading Style Sheets (CSS) is one of the three main components of modern webpages, along with Hypertext Markup Language (HTML) and JavaScript. HTML indicates to the browser what elements should be included on the page (and in what order). CSS indicates how each should be styled. JavaScript provides a means for webpage authors to manipulate these elements programmatically and in response to actions by the end user. Tutorials and reference material covering all three components are available here.

In these pages, we describe CSS further. Text used within HTML, CSS or JavaScript files is generally shown in `courier new` (i.e. a fixed space) font. The pages contain links to an extensive body of reference material explaining HTML, CSS and JavaScript in detail. We also provide a wide range of examples, which can help you understand better how HTML, CSS and JavaScript work. See below for further details on how to access these examples.

CSS instructions can be:

(a) included within an individual HTML element (as part of the mark-up relating to that element), i.e. as 'in-line' CSS
(b) included in the HTML file where the relevant element(s) are located, but not directly within the elements concerned, i.e. as 'in-file' CSS
(c) included in external CSS files, i.e. as 'external' CSS, with a HTML <link> element used to indicate where any such CSS files applicable to a given HTML file are located.

The style attributes of an HTML element can also be altered by JavaScript 'on the fly', e.g. after the page has initially loaded or in response to specific user actions such as clicking a button.

CSS styles typically operate according to a hierarchy, with any JavaScript overrides taking precedence over any CSS styles present when the page is initially loaded but otherwise in-line CSS taking precedence over in-file CSS and in-file CSS taking precedence over external CSS (unless the 'important' characteristic is included in the style statement). In-file CSS is contained in a <style> element. If there is more than one such element then later ones take precedence over earlier ones.

Older versions of HTML (e.g. HTML 4) require <style> elements to be in the `<head>` of the HTML file, although most browsers currently seem to accept them even if they appear in the `<body>`. In theory, the latest HTML version at the time of writing (HTML 5) has the concept of a 'scoped' attribute (e.g. `<style scoped>`) which should allow you to apply different `<style>` elements to different parts of the webpage (which could then legitimately appear in the `<body>` element), but not all browsers currently seem to cater for this aspect of HTML 5.

External style sheets are referenced using a <link> element, which goes inside the `<head>` section. This type of link element has a form such as:

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

External style sheets can be created in any text editor, should not contain any HTML tags (elements) and should be saved with a `.css` extension.

In-file and external CSS are typically set out in the form of 'rule-sets'. A rule set involves a selector and a declaration block. The selector points to the type of HTML element to which the style applies, whilst the declaration block contains one or more style declarations separated by semicolons. Each declaration involves a CSS property name, followed by a colon, followed by the value assigned to the property.

For example, the style rule

```
h3 {color: blue; text-align: center;}
```

has a selector which is `h3` and a declaration block which is `{color: blue; text-align: center;}`. It tells the browser that any `<h3>` element (to which the rule applies) should be centre-aligned and appear in blue. As with HTML, line breaks and multiple spaces are ignored.

Other types of selectors are introduced here and covered in more detail here.

In-line CSS rule-sets involve the style attribute (and do not include a selector or the curly brackets / braces included in in-file or external CSS), e.g. they involve setting the element's style attribute along the lines of: `style = "color: red";`.

Comments in CSS start with `/*` and end with `*/` and can span multiple lines.

Over time CSS has been refined. At the time of writing the latest version is CSS3. Features in CSS1 and CSS2 can typically still be used in CSS3.

Tutorial content

2. Selectors

[CSSTutorialSelectors]

CSS is typically set out in the form of 'rule-sets', which involve a selector and a declaration block. Usually CSS is applied to types of elements. For example, the style rule

```
h3 {color: blue; text-align: center;}
```

has a selector which is `h3` and a declaration block which is `{color: blue; text-align: center;}`. It tells the browser that any `<h3>` element (to which the rule applies) should be centre-aligned and appear in blue. As with HTML, line breaks and multiple spaces are ignored.

However, within HTML you can also define classes of elements with common formatting styles using the element's class attribute. For example, the style rule

```
.center {color: red; text-align: center}
```

would indicate that any element with a class attribute equal to `center` should be centre-aligned and appear in red.

You can also apply CSS to elements of a specific type *and* class. For example, the style rule

```
h3.center {color: green;}
```

would indicate that `<h3>` elements that have their class attribute equal to `center` should be green.

In-file CSS can also be applied to individual elements, if the id attribute of the HTML element has been set (the id attribute should be unique within any given page). If you want to use this type of CSS then precede the id value by a hash (`#`) character.

For example, the style rule

```
#para1 {color: yellow}
```

would be applied to the HTML element with id equal to `para1` (provided there is such an element) and it would appear yellow (unless overridden by a later style rule).

You can also group together rules for elements with the same style definitions, separating each selector with a comma. For example,

```
h1 {color: red;}
h2 {color: red;}
h3 {color: red;}
```

can be grouped together as follows to minimise code and make it easier to follow:

```
h1, h2, h3 {color: red;}
```

More general ways of identifying CSS selectors are set out here.

3. Hints and further information

[CSSTutorialHints]

CSS Values

In CSS, if you are using values that have units, e.g. applying values that are to be interpreted as CSS lengths (e.g. setting the size of an element's left margin using e.g. `margin-left: 20px`) then you should not include a space between the value (here `20`) and the unit (here `px`) as otherwise the style may be ignored.

There are several ways of defining lengths in CSS. There are also specific conventions used when defining CSS times, CSS angles and CSS colours.

Hierarchy in CSS style rules

If you have two or more style rules that would otherwise apply to a specific attribute of a specific element then the hierarchy rules are that:

- More specific rules override more general ones. Specificity is defined based on how many IDs, classes and element names are involved as well as by whether there is an `!important` declaration.
- When even these do not differentiate between styles then whichever one appears last is the one that is applied.

For example, without the `!important` flag, `<h3>` elements using the following styles would appear green (as the green style rule is after the red one), but with the `!important` flag it is the red one that applies in this instance:

```
h3 {color: red !important}
h3 {color: green}
```

Setting the CSS style of the whole page

The style of the whole page can be set by a style rule such as:

```
body {background-color: lightblue;}
```

Multi-valued CSS properties

Some CSS properties take several values. For example, many HTML elements are deemed to have 4 sides (top, right, bottom and left) and there are conventions on how to define properties that encompass all four sides simultaneously, see here.

More generally, some CSS properties are shorthand properties that set several other more granular properties at the same time.

Styling of hyperlinks

Links can be styled differently depending on what state they are:

Link state	Description
a:link	Normal, unvisited link
a:visited	Link that user has visited
a:hover	Link when the user moves a mouse over it
a:active	Link at the moment it is clicked

JavaScript Tutorial

1. Introduction

[JavaScriptTutorialIntroduction]

JavaScript is one of the three main components of modern webpages, along with Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS). HTML indicates to the browser what elements should be included on the page (and in what order). CSS indicates how each should be styled. JavaScript provides a means for webpage authors to manipulate these elements programmatically and in response to actions by the end user. Tutorials and reference material covering all three components are available here.

In these pages, we describe JavaScript further. Text used within HTML, CSS or JavaScript files is generally shown in `courier new` (i.e. a fixed space) font. The pages contain links to an extensive body of reference material explaining HTML, CSS and JavaScript in detail. We also provide a wide range of examples, which can help you understand better how HTML, CSS and JavaScript work. See below for further details on how to access these examples.

JavaScript can be added to a webpage in one of three ways (somewhat akin to how CSS can be added to a webpage):

(a) By including it within an individual HTML event attribute. This typically involves only very small JavaScript statements.
(b) Within separate <script> elements in the HTML
(c) In external script files (these involve including in the HTML a <script> element with its src attribute set to the relevant script file name).

A simple example of JavaScript involves the use of the `document.write` method. For example, the following HTML text would return a web page the first line of which says “Hello World (using HTML)” followed by a line break and a second line saying “Hello World (using HTML)”. Script elements are typically executed in the order in which they appear when the page is first loaded. In this case the script cause the browser to add some more text to the web page.

```
<html>
   <body>
       Hello World (using HTML)<br>
      <script>
         <!--
            document.write("<br>Hello World (using Javascript)")
         //-->
      </script>
   </body>
</html>
```

More sophisticated approaches can alter individual HTML elements rather than merely adding to the end of the document or can react to events such as the clicking of a button. For example, the following HTML text returns a web page with two lines, the first being “Hello World (using HTML)” and the second line being “*and using JavaScript*”.

```
<!DOCTYPE html>
```

```
<html>
<head></head>
<body>
Hello World (using HTML)<br>
<em id="Added"></em>

<script>
document.getElementById("Added").innerHTML="and using JavaScript"
  // Adds text to the element with id="Added"
</script>

</body>
</html>
```

Note: we are here taking advantage of the execution of script commands when the page first loads. A more complicated (but more general way) of achieving the same result would be to add an 'event listener' that is triggered when the page loads and to have a function associated with this event listener that alters (here adds) the text in the desired manner when the event happens. By attaching the function to a different event, e.g. one triggered when the user clicks on an element then the a more responsive webpage can be created.

JavaScript comments

When writing computer software, it often helps to add explanatory comments. In JavaScript, a single line comment is indicated by "*code* // *text*" where the code is still executed, but the text is ignored by the Browser.

Any text between "/*" and "*/" (not in quotes) including line breaks is also ignored, allowing authors to create multi-line comments. These tend to be used for formal documentation, e.g. material at the start of each function that describes what the function does.

Tutorial contents:

2. Variables

[JavaScriptTutorialVariables]

A variable in JavaScript is defined by a command such as:

```
var x;
```

If you want to set a variable to a value when it is first defined then you generally use the assignment operator within this definition, e.g.:

```
var x = 10;
```

JavaScript recognises the following types of 'primitive' variables:

- String variables
- Number variables
- Date variables
- Boolean variables

Variables can also be objects and arrays (and for some string manipulation purposes, regular expressions). In JavaScript, an array is a special type of object that is indexed along the lines of `a[0],a[1]….` Arrays can consist of other objects, including other arrays.

Several variables can be defined in the same statement, with each one separated by a comma, e.g.:

```
var x = 10, y = 15, z = 20;
```

Variables that have not yet been defined a value have their value as `undefined`.

If you redefine a variable, it retains its previous value. For example, after the statements

```
var x = 10;
var x;
```

the variable `x` still has the value `10`.

Variables are manipulated using operators and functions. For example, numbers can be added together using the addition operator or functions can be applied to them, e.g.:

```
var x = 0.1 + 0.2;
function sinsquared(x) {
  var a;
  a = Math.pow(Math.sin(x),2);
  return a;
}
var y = sinsquared(0.3);
```

JavaScript variable names (i.e. identifiers) follow certain rules:

- They can contain only letters, digits, underscores and dollar signs
- They must typically begin with a letter (in some cases they can also begin with a $ or an _ character)
- The names are case sensitive (i.e. a and A are different names)
- They cannot be reserved words, such as those used in JavaScript statement construction)

An important concept in programming is the *scope* of a variable (or more precisely of its name). This is the part of the code within which the relevant variable is accessible via its name. If code is segmented into blocks then it is often desirable to use a similar variable name in different blocks but for the names to then be associated with different variables depending on the block in question. The scope of a JavaScript can be *local* or *global*. Variables defined inside functions are local to that function, whilst those defined outside functions are global in scope. Local variables are deleted when the function completes, while global variables remain available until the user closes the browser window or tab within which the page has been loaded. This means that they are available to new pages loaded into the same browser window. Function arguments work in the same manner as local variables inside a function.

String variables

Strings consist of a series of consecutive characters, e.g.

```
var x = "Cat";
```

A string technically consists of a series (an 'array', except that a JavaScript array is a specific type of variable) of characters, which is zero-indexed. So, if we assigned `x` the value of `"Cat"` then `x[0]` would be `"C"`, `x[1]` would be `"a"`, etc.

Further details on the methods and properties supported by string variables are set out in JavaScript Tutorial: Strings.

Regular expressions

Some string methods and properties involve 'regular expressions'. These take the form:

/pattern/modifiers

e.g.:

```
var x = /cns/i;
```

Further details on the methods and properties supported by regular expressions variables are set out in JavaScript Tutorial: Regular Expressions.

Numbers (and mathematical manipulations)

JavaScript has only one type of number (in contrast to, e.g. Visual Basic, which differentiates between e.g. integers, floating point numbers and double precision numbers). Numbers can be written with or without decimal points and/or with or without (scientific) exponents), e.g.

```
var x = 4.1;     // With a decimal point
var y = 4;       // Without a decimal point
```

```
var p = 135e6   // Means 135000000
var q = 13.5e-3 // Means 0.0135
```

Further details on the methods and properties supported by numbers and by the Math object (which can be used to carry out mathematical manipulations) are set out in JavaScript Tutorial: Number variables and mathematical functions.

Booleans

Boolean variables take one of two values, `true` or `false`. They are instantiated by a statement such as:

```
var b = true;
```

You can use the `Boolean()` function to identify whether an expression is true or false, although it is simpler just to use operators that return Boolean outputs, e.g. `Boolean(2 > 1)`, `(2 > 1)` or even `2 > 1` all return true.

Further details on the methods and properties supported by Boolean variables are shown in JavaScript Tutorial: Booleans.

Arrays

Arrays contain multiple (indexed) values in a single variable. Array indices are zero-based, i.e. the first element of the array has as its index 0, the second 1 etc. They are instantiated by statements such as:

```
var a = ["France", "Germany"];
var b = [1, 2, 5, 4];
```

It is worth noting that elements of arrays can themselves be arrays since technically an array is a specific type of object.

Further details on the methods and properties supported by arrays (and some of the subtleties that arise if you want to copy them) are set out in JavaScript Tutorial: Arrays.

Objects

JavaScript objects are containers that contain *properties* and *methods*. For example, a statement such as:

```
var person = {title:"Mr", surname:"Smith", age:30}
```

creates an object that has three properties, i.e. name-value, pairs that in this instance characterise (some of the features of) a person.

Object properties can be accessed in two ways, either here e.g. `person.title` or `person["title"]` (both of which in this instance would return a value of `"Mr"`). An array is a specific type of object with the property names indexed from 0 up to the length of the array less 1 (and hence elements of arrays can themselves be arrays or other sorts of objects).

Object methods are functions that can be applied to objects. They are technically also property-like in nature, i.e. again come in name-value pairs, but with the 'name' being a function name (with parameter definitions if necessary) and the 'value' being the JavaScript function script associated with that function, see JavaScript Tutorial: Objects.

A special type of object is the Error object, which is used for error handling.

3. Statements

[JavaScriptTutorialStatements]

JavaScript statements identify instructions that are executed by the web browser. For example, the following statement tells the browser to write "Hello World" inside an HTML statement with the id attribute = `"element"`:

```
document.getElementById("element").innerHTML = "Hello World"
```

The same result can be achieved using several separate statements, e.g.:

```
var d = document.getElementById("element");
var x = "Hello";
var y = " World";
var z = x + y;
d.innerHTML = z;
```

Statements are separated by semicolons and multiple statements are allowed on one line. JavaScript ignores multiple spaces (except in strings, i.e. within quotation marks). A common good practice is to put spaces around operators (e.g. =, + , …). Very long lines of code are also often frowned upon, and are usually broken after an operator.

Statements can (and often are) grouped together in code blocks, inside curly brackets, i.e. { … }. A particularly important example of the use of code blocks involves functions, which provide a means of executing on demand one or more statements, e.g.:

```
function func() {
      document.getElementById("element").innerHTML = "Hello";
}
```

Statements often start with a statement identifier. These are reserved words which cannot be used as variable names or for other purposes. A list of statement reserved words recognised by JavaScript is shown here. They include: `break`, `continue`, `do`, `for`, `if`, `return`, `switch`, `throw`, `try`, `catch`, `var` and `while`.

Most JavaScript programs contain many statements, which are executed one by one in the order in which they are written except when statement flow control is adjusted using statements such as `for`, `if` or `while`.

4. Functions

[JavaScriptTutorialFunctions]

A JavaScript function is a block of JavaScript code that can be executed as a discrete unit. It involves a function statement along the lines of e.g.:

```
function func() {
      document.getElementById("element").innerHTML = "Hello";
}
```

Function definitions can include parameters (separated by a comma if more than one parameter), e.g. the following (if passed a string variable) would allow any text to be inserted in the relevant element's innerHTML.

```
function func2(x) {
      document.getElementById("element").innerHTML = x;
}
```

Such a function would be invoked by JavaScript such as `func2("Hello World")`.

Functions are much like procedures or subroutines in other programming languages. The code inside the curly brackets executes when the function is invoked. This can happen when an event occurs, when the function is called from JavaScript code or sometimes when it is self-invoked. If a function includes a return statement then the function will stop executing and will return the value identified by the function's return statement. The function (technically, a special type of object) can be distinguished from the act of invoking it. The `()` operator invokes the function, e.g. in the above `func` refers to the function object, but `func()` will invoke the function itself.

The function parameters are the names listed in the function definition (i.e. the `x` in the definition of `func2`). Function arguments are the values received by the function (i.e. assigned to the function parameters) when it is invoked.

Function names can contain letters, digits, underscores and dollar signs (the same rules as apply to variable naming applies to function naming). Wherever a variable can be used, a valid function call evaluating to the same value can also be used.

5. Event handling

[JavaScriptTutorialEventHandling]

A responsive website needs to respond to users when the users act in specific ways, e.g. loading a page, clicking on a button, moving a mouse around a document window etc. JavaScript, like many other modern more sophisticated general-purpose programming languages, includes the concept of *events*. These assign specific functions to specific events, with the functions being invoked if/when the event occurs.

Event handling linked to individual elements, such as what happens when someone clicks on an element, is often implemented by assigning a specific function to the event attribute of that element, see here.

Global events (not linked to specific HTML elements), such as those triggered by loading the page, are typically implemented by using e.g. the document.addEventListener method, e.g.:

```
document.addEventListener('load', addtext());
```

6. The Document Object Model (DOM)

[JavaScriptTutorialDOM]

The JavaScript HTML Document Object Model ('DOM') provides a way for JavaScript to access all elements of an HTML webpage. Fuller details of the DOM are given here. There is an associated Browser Object Model (BOM), details of which are given here.

The browser creates a DOM (i.e. a `document` object) for a page when the page is first loaded. This has a tree structure, with the root element (or 'node') being the page's <html> element. The root element then has two sub-elements (or 'nodes'), i.e. the <head> element and the <body> element.

The <head> element will in turn often include as one of its 'child' nodes a <title> element. The <body> element contains the main body of the webpage and will typically contain many different elements, some nested within others. JavaScript can change all the elements (including all their attributes), can add new elements or remove existing ones, can react to all existing events and create new ones.

Formally, the DOM is a W3C (World Wide Web Consortium) standard. It has the following aim, according to W3C: "*The W3C Document Object Model (DOM) is a platform and language-neutral interface that allows programs and scripts to dynamically access and update the content, structure and style of a document.*" It is formally subdivided into three parts: (a) the Core DOM for all document types, (b) the XML DOM for XML documents, and (c) the HTML DOM for HTML documents. It adopts an object-orientated programming approach, with the HTML elements being *objects* which have *properties* and to which can be applied *methods*. The elements can also trigger *events*.

A common way of accessing an element is to assign it an id (i.e. set its id attribute object to a prespecified value). The element can then be identified in Javascript by applying the `getElementById` method to the document object, returning an object corresponding to the element. Its properties (e.g. its innerHTML property, which is the text within an element) can be set by assigning values to the relevant property of the element object. For example, the following example returns a web page that says "hello".

```
<html>
<body>
```

```
<div id="example"></div>
<script>document.getElementById("example").innerHTML =
"hello"</script>
</body>
</html>
```

Common ways of accessing the DOM include:

Aim	Example JavaScript	Description
Finding / accessing elements	`document.getElementById(`*id*`)`	Returns object corresponding to element with this id attribute
	`document.getElementsByTagName(`*name*`)`	Returns collection of elements by tag name (i.e. by type of element)
	`document.getElementsByClassName(`*name*`)`	Returns collection of elements by class name
	`document.querySelectorAll(`*CSSSelector*`)`	Returns collection of elements by CSSSelector
Changing elements	*element*`.innerHTML =` *HTMLcontent*	Change inner HTML of element
	element.attribute `=` *value*	Change attribute value of element (value needs to be valid for that attribute)
	element`.setAttribute(`*attribute*`,` *value*`)`	Change attribute value for a given element
Adding and deleting elements	`document.createElement(`*element*`)`	Creates HTML element
	`document.appendChildElement(`*element*`)`	Add HTML element
	`document.removeChildElement(`*element*`)`	Remove HTML element
	`document.replaceChildElement(`*element*`)`	Remove HTML element
	`document.write(`*element*`)`	Write directly to HTML output

Other points to note about the DOM include:

(a) The DOM uses the idea of nodes and a node tree. This involves a tree structure where each branching point is a separate node. So, nodes belong to (are children of) just one other node (their parent) back to the root node (which in the DOM is the `document` object)
(b) HTML elements are 'element' nodes, the attributes of these elements are 'attribute' nodes, the text within HTML elements are 'text' nodes and comments are 'comment' nodes
(c) A NodeList object represents a set of nodes, e.g. an HTML element's collection of child nodes. These will be indexed and each node within the NodeList can then be associated with another NodeList (its children)
(d) The HTML document, once it has been loaded into the web browser, is formally part of the corresponding Window object and can therefore be accessed via `window.document`
(e) The DOM supports a range of (own) methods and properties, see here.
(f) HTML elements ('nodes') within the DOM also support a range of more generic methods and properties, see here. These also apply to the `document` object itself but do not always make much sense when applied in this manner.

(g) HTML element attributes are represented by an Attr object. These are always children of a specific HTML element. The properties and methods that apply to Attr objects are shown here.
(h) A NamedNodeMap object represents an unordered collection of nodes, e.g. the set of attributes assigned to a given HTML element. Properties and methods that apply to NamedNodeMap objects are shown here.
(i) The DOM object and its components can be thought of as an example of an XML document. XML documents have several methods and properties not otherwise covered in the above (such as XMLHTTPRequest, which can be used to send, request and receive data from the server once a webpage has loaded), see here.

Further details are set out in the following pages and in links within them:

1. DOM own properties and methods
2. HTML Element objects: Properties and Methods
3. HTML Attribute objects: Properties and Methods
4. NamedNodeMap objects: Properties and Methods
5. Event objects: Properties and Methods
6. MouseEvent objects: Properties and Methods
7. KeyboardEvent objects: Properties and Methods
8. HashChangeEvent objects: Properties and Methods
9. PageTransitionEvent objects: Properties and Methods
10. FocusEvent objects: Properties and Methods
11. AnimationEvent objects: Properties and Methods
12. TransitionEvent objects: Properties and Methods
13. WheelEvent objects: Properties and Methods
14. TouchEvent objects: Properties and Methods

I. Style objects: Properties and Methods
II. Creating and Accessing HTML Elements in JavaScript
III. Standard HTML DOM properties and methods
IV. The JavaScript BOM (Browser Object Model)
V. The JavaScript XML DOM

Formatting elements

[HTMLElementsFormatting]

The following is a list of HTML formatting elements:

Tag	Description	More	Further comments
<abbr>	Abbreviation or acronym	Here	
<acronym>	Acronym	Here	Not supported in HTML 5
<address>	Contact information for the author or owner of a document	Here	
<b>	Bold text	Here	
<basefont>	Default font, colour and size of all text in a document	Here	Not supported in HTML 5 (instead use CSS)
<bdi>	Isolates a part of text that might be formatted in a different direction to other text outside that part	Here	New in HTML 5

<bdo>	Overrides the current text direction	Here	
<big>	Big text	Here	Not supported in HTML 5 (instead use CSS)
<blockquote>	Section that is quoted from another source	Here	
	Centred text	Here	Not supported in HTML 5 (instead use CSS)
<cite>	Title of a work	Here	
<code>	A piece of computer code	Here	
<del>	Indicates text deleted from a document	Here	
<dfn>	Defining instance of a term	Here	
	Emphasised text	Here	Often used to italicise text, but ideally this should be done using CSS
<font>	Font, colour and size of text	Here	Not supported in HTML 5 (instead use CSS)
<h1>, <h2>, <h3>, <h4>, <h5>, <h6>	HTML headings	Here	Provides a hierarchy of headings

<i>	A part of text in an alternate voice or mood	Here	
<ins>	Indicates text added to a document	Here	
<kbd>	Keyboard input	Here	
<mark>	Marked/highlighted text	Here	New in HTML 5
<meter>	A scalar measurement within a specific range (a gauge)	Here	New in HTML 5
<p>	Paragraph	Here	
<pre>	Preformatted text	Here	
<progress>	Represents the progress of a task	Here	New in HTML 5
<q>	Short quotation	Here	
<rp>	Indicates what to show in browsers that do not support ruby annotations	Here	New in HTML 5
<rt>	Explanation / pronunciation of characters	Here	New in HTML 5. For East Asian typography
<ruby>	Ruby annotation	Here	New in HTML 5. For East Asian typography
<s>	Text that is no longer correct	Here	
<samp>	Sample output from a computer program	Here	
<small>	Smaller text	Here	
<strike>	Strikethrough text	Here	Not supported in HTML 5 (instead use <del> or <s>)
	Defines more important text	Here	Commonly used as another way of highlighting text or making it bold
<sub>	Subscripted text	Here	
<sup>	Superscripted text	Here	
<time>	Date / time	Here	New in HTML 5
<tt>	Teletype text	Here	Not supported in HTML 5 (instead use CSS)
<u>	Text that should be stylistically different from normal text	Here	Commonly used for underlining
<var>	Variable	Here	
<wbr>	Posible line-break	Here	New in HTML 5

The default styles applicable to these elements are shown here. The behaviour of most formatting elements can be replicated using CSS.

<abbr>

[HTMLElementAbbr]

The HTML <abbr> element indicates an abbreviation or acronym. The full text which is being abbreviated can be included in the element's title attribute and it will then typically appear as tooltip text if the mouse is moved over the element.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<address>

[HTMLElementAddress]

The HTML `<address>` element is usually used to define contact information for the author or owner of a document or file. If it is inside an <article> element then it typically represents contact information for that article. If it is outside an article element but inside a <body> element then it typically represents contact information for the document or page.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<area>

[HTMLElementArea]

The HTML `<area>` element identifies an area inside an image-map.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`alt`	Specifies alternative text to show when original content fails to display	Here
`coords`	Specifies the coordinates of an <area>	Here
`download`	Target will be downloaded when user clicks on hyperlink	Here
`href`	URL of page the link goes to	Here
`hreflang`	Language of linked document	Here
`media`	Specifies media / device linked document is optimised for	Here
`rel`	Relationship between current document and linked document	Here
`shape`	Specifies shape of an <area> element	Here
`target`	Specifies where / how to open the linked document (or where to submit the form)	Here
`type`	Type of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above (except the `download`, `hreflang`, `media`, `rel` and `type` attribute). The corresponding HTML DOM object also typically supports the `hash`, `host`, `hostname`, `origin`, `password`, `pathname`, `port`, `protocol`, `search` and `username` variants of the `href` attribute, see here for more details. The default style applicable to this element is shown here.

<aside>

[HTMLElementAside]

The HTML `<aside>` element indicates some content separate from but related to surrounding context. It is new in HTML 5.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<audio>

[HTMLElementAudio]

The HTML `<audio>` element is used to define and play sound, such as music or other audio streams. Supported file formats include MP3 (nearly all browsers), Wav (not Internet Explorer) and Ogg (some browsers). It is new in HTML 5.

If the browser does not support `<audio>` elements then any text between the `<audio>` and `</audio>` tags will be displayed.

The attributes it can take (in addition to HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`autoplay`	Specifies whether media should start playing as soon as ready	Here
`controls`	Whether controls (such as play and pause buttons) should be displayed	Here
`loop`	Media to start over again when it finishes	Here
`muted`	Audio output should be muted	Here
`preload`	If / how author thinks media should be loaded when page loads	Here
`src`	URL of media	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports DOM generic media properties and methods and the following additional properties and methods.

Additional methods:

Method	Description	More
`fastSeek()`	Seeks to a specified time in audio	Here
`getStartDate()`	Returns Date object representing current timeline offset	Here

The default style applicable to this element is shown here.

<b>

[HTMLElementB]

The HTML <b> element indicates bold text. According to the HTML 5 specification it should be used as a last resort when no other elements such as , <h1>, <h2>, <h3>, <h4>, <h5> or <h6> are appropriate.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<base>

[HTMLElementBase]

The HTML <base> element specifies the base URL for all relative URLs in a document. There can be at most one <base> element in any specific document (and it should be inside the relevant <head> element).

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
href	URL of page the link goes to	Here
target	Specifies where / how to open the linked document (or where to submit the form)	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also typically supports the hash, host, hostname, origin, password, pathname, port, protocol, search and username variants of the href attribute, see here for more details. The default style applicable to this element is shown here.

<basefont>

[HTMLElementBasefont]

The HTML <basefont> element was used to indicate the default font, colour and size of all text in a document. It is not supported in HTML 5. Instead, use CSS.

<bdi>

[HTMLElementBdi]

The HTML <bdi> element indicates material that might be formatted in a different direction from other material surrounding the element. 'bdi' stands for 'bi-directional isolation'. It is new in HTML 5. A similar effect can usually be achieved using the unicode-bidi style applied to e.g. a span element, but the semantic meaning is only conveyed by the <bdi> element and in some cases browsers may ignore CSS, but not a <bdi> element.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<bdo>

[HTMLElementBdo]

The HTML <bdo> element makes it possible to override the current text direction.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
dir	Text direction for element content	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. The default style applicable to this element is shown here.

<big>

[HTMLElementBig]

The HTML <big> element was used to indicate big text. It is not supported in HTML 5. Instead, use CSS.

<blockquote>

[HTMLElementBlockquote]

The HTML <blockquote> element indicates a section that is quoted from another source. Browsers often indent text in such elements (typically a <q> element is used for an in-line quotation and isn't indented).

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
cite	URL which explains the quote / deleted / inserted text	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. The default style applicable to this element is shown here.

<body>

[HTMLElementBody]

The HTML <body> element identifies the body of the document and contains all the visible contents of the document, such as text, hyperlinks, tables and images etc.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the alink, background, bgcolor, link, text and vlink attributes, but these are no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

[HTMLElementBr]

The HTML <br> element indicates a single line break (c.f. a carriage return). In XHTML, it needs to be 'properly' closed as per <br />.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<button>

[HTMLElementButton]

The HTML <button> element indicates a clickable button. Inside a <button> element (unlike an <input> element) you can put content such as text or images. You should typically specify the type attribute for <button> element as different browsers do not necessarily default to the same type. Within a <form> element you should also bear in mind that different browsers may submit different values.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
autofocus	Specifies whether element should automatically get focus when page loads	Here
disabled	Specified element(s) to be disabled	Here
form	Name of the form that element belongs to	Here
formaction	Where to send form-data to when form submitted	Here. Only for type = submit
formenctype	How form-data should be encoded before sending it to a server	Here. Only for type = submit
formmethod	How to send form-data (i.e. which HTTP method to use)	Here. Only for type = submit

formnovalidate	Specifies that form-data should not be validated on submission	Here. Only for type = submit
formtarget	Specifies where to display the response that is received after submitting form	Here. Only for type = submit
name	Name of element	Here
type	Type of element	Here
value	Value of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. However, the DOM versions of formaction, formenctype, formmethod, formnovalidate and formtarget need to adopt the JavaScript name capitalisation convention, i.e. need to be: `formAction`, `formEnctype`, `formMethod`, `formNoValidate` and `formTarget` respectively. The default style applicable to this element is shown here.

<form>

[HTMLElementForm]

The HTML `<form>` element indicates an HTML form for user input. Typically, a `<form>` element will contain one or more of the following form elements:

- <button>
- <fieldset>
- <input>
- <label>
- <optgroup>
- <option>
- <select>
- <textarea>

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`accept-charset`	Specifies character encodings used for form submission	Here
`action`	Where to send form-data when form submitted	Here
`autocomplete`	Specifies whether element has autocomplete enabled	Here
`enctype`	How form-data to be encoded when submitted to server	Here. Only for method = `post`
`method`	Specifies HTTP method used when sending form-data	Here
`name`	Name of element	Here
`novalidate`	Form should not be validated when submitted	Here
`target`	Specifies where / how to open the linked document (or where to submit the form)	Here

It also used to take the accept attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above (with the `novalidate` property of the underlying element corresponding to the `noValidate` property of the DOM object). It also supports the following additional properties and methods:

Additional properties:

Property	Description	More
`encoding`	An alias for `enctype`	Here
`length`	Returns number of elements in form	Here

Additional properties:

Method	Description	More
`reset()`	Resets form	Here
`submit()`	Submits form	Here

The default style applicable to this element is shown here.

<frame>

[HTMLElementFrame]

The HTML `<frame>` element was used to indicate a window (frame) within a frameset. It is not supported in HTML 5.

<h1>

[HTMLElementH1]

The HTML `<h1>` element indicates a level 1 HTML heading.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<h2>

[HTMLElementH2]

The HTML `<h2>` element indicates a level 2 HTML heading.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<h3>

[HTMLElementH3]

The HTML `<h3>` element indicates a level 3 HTML heading.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<h4>

[HTMLElementH4]

The HTML `<h4>` element indicates a level 4 HTML heading.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<h5>

[HTMLElementH5]

The HTML `<h5>` element indicates a level 5 HTML heading.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<h6>

[HTMLElementH6]

The HTML `<h6>` element indicates a level 6 HTML heading.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<head>

[HTMLElementHead]

The HTML `<head>` element provides information about the document. The `<head>` element can contain the following sorts of elements (it is always supposed to include a <title> element, but HTML that does not do so may not be rejected by browsers):

- <base>
- <link>
- <meta>
- <noscript>
- <script>
- <style>
- <title>

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `profile` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<header>

[HTMLElementHeader]

The HTML `<header>` element indicates a header for a document or section. It is new in HTML 5. Typically, a `<header>` element might contain introductory content, navigational links, one or more heading elements (i.e. <h1>, <h2>, <h3>, <h4>, <h5> or <h6>), a logo and perhaps also some authorship information. A document can contain several `<header>` elements.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

headings

[HTMLElementHeadings]

The HTML <h1>, <h2>, <h3>, <h4>, <h5> and <h6> elements provide a hierarchy of headings (with the <h1> element by default having the largest size and the <h6> element the smallest size).

<hr>

[HTMLElementHr]

The HTML `<hr>` element indicates a thematic change in the content. Often it is rendered as a line across the window.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align`, `noshade`, `size` and `width` attributes, but these are no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<html>

[HTMLElementHtml]

The HTML `<html>` element is the root node of an HTML document. Only <!DOCTYPE> elements should appear outside it. It tells the browser that the document is an HTML document.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
manifest	Specifies address of document's cache manifest (for offline browsing)	Here
xmlns	Indicates the XML namespace attribute (if the content needs to conform to XHTML); is not an HTML attribute as such and is added by default if needed	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<i>

[HTMLElementI]

The HTML <i> element indicates a part of text in an alternate voice or mood. Typically it is rendered as italicised text. Convention recommends using the <i> element only when there isn't a more appropriate element, such as <cite>, <dfn>, , <mark> or .

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<iframe>

[HTMLElementIframe]

The HTML <iframe> element indicates an inline frame. It can also be used to embed another document within the current HTML document.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
height	Height of element	Here
name	Name of element	Here
sandbox	Allows an extra set of restrictions for the content of an <iframe> element	Here
src	URL of resource	Here
srcdoc	HTML content of page to show in an <iframe>	Here
width	Width of element	Here

It used to support the align, frameborder, longdesc, marginheight, marginwidth and scrolling attributes, but these are no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports the following additional properties:

Property	Description	More
`contentDocument`	Returns document object generated by the iframe	Here
`contentWindow`	Returns window object generated by the iframe	Here

The default style applicable to this element is shown here.

[HTMLElementImg]

The HTML `<img>` element indicates an image. It technically has two required attributes, namely `src` (the source of the image) and `alt` (the alternative text displayed if the image cannot be found or cannot be displayed by the browser), although the `alt` attribute can typically be dispensed with. Images are not technically inserted into an HTML page but instead are linked to the page. The `<img>` element therefore creates a placeholder that will include the image once the page is rendered (and the image retrieved by the rendering process from its source location).

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`alt`	Specifies alternative text to show when original content fails to display	Here
`crossorigin`	Specifies how element handles cross-origin requests	Here
`height`	Height of element	Here
`ismap`	Image is a server-side image-map	Here
`sizes`	Specifies size of linked resource	Here
`src`	URL of resource	Here
`srcset`	URL of image to use in different situations	Here
`usemap`	Specifies an image as a client-side image-map	Here
`width`	Width of element	Here

It used to support the `align`, `border`, `hspace`, `longDesc` and `vspace` attributes, but these are no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above (with the `crossorigin`, `ismap` and `usemap` properties of the underlying element corresponding to the `crossOrigin`, `isMap` and `useMap` properties of the DOM object). It also supports the following additional properties:

Property	Description	More
`complete`	Returns whether the browser has finished loading the image	Here
`naturalHeight`	Returns original height of image	Here

naturalWidth	Returns original width of image	Here

The default style applicable to this element is shown here.

<input>

[HTMLElementInput]

The HTML <input> element indicates a (single-line) input control into which the user can enter data. It is used within a <form> element. There are many different types of <input> element that vary depending on the type attribute of the element, including:

- button, checkbox, color, date, datetime, datetime-local, email, file, hidden, image, month, number, password, radio, range, reset, search, submit, tel, text, time, url, week

In HTML markup <input> elements are empty (they only involve attributes). Labels for input elements can be defined using the <label> element.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
accept	Specifies types of file accepted by server	Here. Only for type = file
alt	Specifies alternative text to show when original content fails to display	Here
autocomplete	Specifies whether element has autocomplete enabled	Here
autofocus	Specifies whether element should automatically get focus when page loads	Here
checked	Specifies that the element should be pre-selected	Here. For type= checkbox or type = radio
dirname	Specifies text direction will be submitted	Here
disabled	Specified element(s) to be disabled	Here
form	Name of the form that element belongs to	Here
formaction	Where to send form-data to when form submitted	Here. Only for type = image and type = submit
formenctype	How form-data should be encoded before sending it to a server	Here. Only for type = image and type = submit
formmethod	How to send form-data (i.e. which HTTP method to use)	Here. Only for type = image and type = submit
formnovalidate	Specifies that form-data should not be validated on submission	Here. Only for type = image and type = submit
formtarget	Specifies where to display the response that is received after submitting form	Here. Only for type = image and type = submit
height	Height of element	Here

list	Refers to <datalist> that contains pre-defined options	Here
max	Maximum value	Here
maxlength	Maximum number of characters allowed in an element	Here
min	Minimum value	Here
multiple	Indicates that a user can enter more than one value	Here
name	Name of element	Here
pattern	Regular expression that value of element is checked against	Here
placeholder	Short hint describing expected value of element	Here
readonly	Whether element is read-only	Here
required	Whether the element must be filled out before submitting form	Here
size	Specifies width in characters of element	Here
src	URL of resource	Here
step	Accepted number intervals	Here
type	Type of element	Here
value	Value of element	Here
width	Width of element	Here

Some of these attributes are valid for only some <input> element types:

type	Valid attributes
all	autofocus (except for hidden type), disabled (except for hidden type), form, name, type, value
button	-
checkbox	checked, required
color	autocomplete, list
date	autocomplete, max, min, readonly, required, step
datetime	autocomplete, list, max, min, readonly, required, step
datetime-local	autocomplete, list, max, min, readonly, required, step
email	autocomplete, list, maxlength, multiple, pattern, placeholder, readonly, required, size
file	accept, multiple, required
hidden	-
image	alt, formAction, formEnctype, formMethod, formNoValidate, formTarget, height, src, width
month	autocomplete, list, max, min, readonly, required, step
number	autocomplete, list, max, min, placeholder, readonly, required, step
password	autocomplete, maxlength, pattern, placeholder, readonly, required, size
radio	checked, required
range	autocomplete, list, max, min, step
reset	-
search	autocomplete, list, maxlength, pattern, placeholder, readonly, required, size
submit	formAction, formEnctype, formMethod, formNoValidate,

	formTarget
text	autocomplete, list, maxlength, pattern, placeholder, readonly, required, size
time	autocomplete, list, max, min, readonly, required, step
url	autocomplete, list, maxlength, pattern, placeholder, readonly, required, size
week	autocomplete, list, max, min, readonly, required, step

It used to accept the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above (with the `formaction`, `formenctype`, `formmethod`, `formnovalidate`, `formtarget`, `maxlength` and `readonly` properties of the underlying element corresponding to the `formAction`, `formEnctype`, `formMethod`, `formNoValidate`, `formTarget`, `maxLength` and `readOnly` properties of the DOM object). It also supports the following additional properties and methods:

Additional properties:

Property	Description	Applies to type
`defaultChecked`	Returns default value of checked attribute	`checkbox`, `radio`. See Here
`defaultValue`	Sets / returns default value	*All*. See Here
`files`	Returns a FileList object representing file(s) selected by upload button	`file`. See Here
`form`	Returns form that contains element	*All*. See Here
`indeterminate`	Sets / returns value of its indeterminate status	`checkbox`. See Here

Additional methods:

Method	Description	Applies to type
`select()`	Selects content of password field	`Password`. See Here
`stepDown()`	Decrements value of relevant field by specified amount	`datetime`, `datetime-local`, `month`, `number`, `range`, `time`, `week`. See Here
`stepUp()`	Increments value of relevant field by specified amount	`datetime`, `datetime-local`, `month`, `number`, `range`, `time`, `week`

The default style applicable to this element is shown here.

<optgroup>

[HTMLElementOptgroup]

The HTML `<optgroup>` element indicates a group of related options in a drop-down list.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`disabled`	Specified element(s) (here the option-group) to be disabled	Here
`label`	Title of option group	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above). The default style applicable to this element is shown here.

<option>

[HTMLElementOption]

The HTML `<option>` element indicates an option in a drop-down list.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`disabled`	Specified element(s) (here the option-group) to be disabled	Here
`label`	Title of option group	Here
`selected`	Indicates that an <option> element should be pre-selected when the page loads	Here
`value`	Value of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports the following additional properties:

Property	Description	More
`defaultSelected`	Returns default value of the selected attribute	Here
`form`	Returns reference to form that contains the option	Here
`index`	Sets / returns index position of option in drop-down list	Here
`text`	Sets / returns text of the option	Here

The default style applicable to this element is shown here.

<output>

[HTMLElementOutput]

The HTML `<output>` element indicates the result of a calculation (e.g. one performed by a script). It is new in HTML 5.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`for`	Specifies which form element(s) a label calculation is bound to	Here
`form`	Name of the form that element belongs to	Here
`name`	Name of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above (with the `for` property

of the underlying element corresponding to the `htmlFor` property of the DOM object). It also supports the following additional properties:

Property	Description	More
`defaultValue`	Sets / returns default value	Here
`labels`	Returns a collection of <label> elements associated with the `<output>` object	Here
`type`	Returns type of HTML element represented by the `<output>` object	Here
`value`	Sets / returns value of element	Here

The default style applicable to this element is shown here.

<p>

[HTMLElementP]

The HTML `<p>` element indicates a paragraph.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align` attribute, but this is no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<param>

[HTMLElementParam]

The HTML `<param>` element indicates a parameter for an object.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`name`	Name of associated element	Here
`value`	Value of element	Here

It used to support the `type` and `valuetype` attributes, but these are no longer supported in HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above). The default style applicable to this element is shown here.

<picture>

[HTMLElementPicture]

The HTML <picture> element indicates a container for multiple image resources. A <picture> element contains zero or more <source> elements followed by one <img> element. The source element(s) will be differentiated by different srcset attributes (required, defines the URL of the image to be shown by the <picture> element) and by the media attribute (optional, a CSS media query that identifies which media relates to that URL). The browser uses the first matching <source> element, and if none match then it defaults to the <img> element.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<pre>

[HTMLElementPre]

The HTML <pre> element indicates a piece of preformatted text. Typically the text is displayed in a fixed-width font (usually Courier), preserving both spaces and line breaks.

The attributes it can take are HTML global attributes and HTML event attributes. It used to support the width attribute, but this is no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<progress>

[HTMLElementProgress]

The HTML <progress> element is most commonly used to show the progress of some task. It is new in HTML 5. It is not very suitable for representing a gauge (like a fuel tank), for which better usually is to use a <meter> element.

For example, markup as follows:

```
Progress so far: <progress value="40" max="100">
```

creates output that involves a progress bar showing that 40% of the task has been completed:

If you want the bar to be narrower than it is by default then you need to use the width attribute within the style of the element, e.g. markup as follows:

```
Progress so far: <progress value="33" max="100" style="width:40px">
```

Usually a progress bar will be updated as time progresses, often using JavaScript.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
max	Maximum value	Here
value	Value of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports the following additional properties:

Property	Description	More
labels	Returns a list of the progress bar labels (if any)	Here
position	Returns current position of progress bar	Here

The default style applicable to this element is shown here.

<q>

[HTMLElementQ]

The HTML <q> element indicates a short quotation.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
cite	URL which explains the quote	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. The default style applicable to this element is shown here.

<rp>

[HTMLElementRp]

The HTML <rp> element indicates what to show in browsers that do not support ruby annotations (for East Asian typography). It is new in HTML 5.

It is used in conjunction with <rt> and <ruby> elements (the <ruby> element includes one or more characters that need an explanation / pronunciation, the <rt> element gives that information and the optional <rp> element indicates what to show for browsers that do not support such characters.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<rt>

[HTMLElementRt]

The HTML <rt> element indicates an explanation / pronunciation of characters (typically for East Asian typography). It is new in HTML 5.

It is used in conjunction with <rp> and <ruby> elements (the <ruby> element includes one or more characters that need an explanation / pronunciation, the <rt> element gives that information and the optional <rp> element indicates what to show for browsers that do not support such characters.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<ruby>

[HTMLElementRuby]

The HTML <ruby> element indicates ruby annotation (for East Asian typography). It is new in HTML 5.

It is used in conjunction with <rp> and <rt> elements (the <ruby> element includes one or more characters that need an explanation / pronunciation, the <rt> element gives that information and the optional <rp> element indicates what to show for browsers that do not support such characters.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<s>

[HTMLElementStrikeThrough]

The HTML <s> element indicates text that is no longer correct. Conventionally, the <s> element should not be used for replaced or deleted text (instead use a <del> element).

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<samp>

[HTMLElementSamp]

The HTML <samp> element is a phrase element indicating sample output from a computer program. It is not depreciated, but typically a richer effect can be achieved using CSS.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<script>

[HTMLElementScript]

The HTML <script> element indicates client-side script / programming code. Usually this is written in JavaScript. The <script> element either contains this code or points to an external file via its src attribute (if the src attribute is present then the <script> element must be empty). The contents of an noscript element indicates what happens for users who have disabled scripts in their browser or whose browser does not support client-side scripting.

The way in which a script executes is driven by the async and defer attributes. If neither are present then the script is fetched and executed immediately the browser reaches the element (before the browser continues parsing the page).

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
async	Indicates if script to be executed asyncronously	Here. Only for external scripts
charset	Specifies character encoding	Here
defer	Script to be executed only when page has finished parsing	Here. Only for external scripts
src	URL of resource	Here
type	Type of element	Here

It used to support the xml:space attribute, but this is no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports the following additional properties:

Property	Description	More
crossOrigin	Sets / returns the CORS settings for the script	Here
text	Sets / returns contents of all child text nodes of the script	Here

The default style applicable to this element is shown here.

<section>

[HTMLElementSection]

The HTML `<section>` element indicates a section in a document (e.g. a discrete chapter / heading / footer etc>. It is new in HTML 5.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<select>

[HTMLElementSelect]

The HTML `<select>` element indicates a drop-down list. The option elements within the `<select>` element identify the available options within the drop-down list

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`autofocus`	Specifies whether element should automatically get focus when page loads	Here
`disabled`	Specified element(s) to be disabled	Here
`form`	Name of the form that element belongs to	Here
`multiple`	Indicates that a user can enter more than one value	Here
`name`	Name of element	Here
`required`	Whether the element must be filled out before submitting form	Here
`size`	Specifies number of visible options	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports the following additional properties and methods:

Additional properties:

Property	Description	More
`length`	Returns number of option elements within the drop-down list	Here
`options`	Returns a collection of all options in drop-down list	Here
`selectedIndex`	Sets / returns index of selected option	Here
`type`	Returns type of form the drop-down list is within	Here
`value`	Sets / returns the value of the selected option in the drop-down list	Here

Additional methods:

Method	Description	More
`add()`	Adds an option to drop-down list	Here
`remove()`	Removes an option from drop-down list	Here

The default style applicable to this element is shown here.

<small>

[HTMLElementSmall]

The HTML `<small>` element indicates smaller text.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<source>

[HTMLElementSource]

The HTML `<source>` element allows multiple media resources for media elements. It links together associated <video> and <audio>. It is new in HTML 5. The srcset attribute is required if the `<source>` element is used in a picture element, whilst the src attribute is required when the `<source>` element is used in an <audio> or <video> element.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`media`	Specifies media / device linked document is optimised for	Here
`sizes`	Specifies image size(s)	Here
`src`	Required when URL of resource	Here
`srcset`	URL of image to use in different situations	Here
`type`	Type of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above). The default style applicable to this element is shown here.

[HTMLElementSpan]

The HTML `<span>` element indicates a section in a document. It is usually defined with its own style.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

[HTMLElementStrong]

The HTML `<strong>` element is a phrase element indicating more important text. It is commonly used as a way of highlighting text or making it bold.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<style>

[HTMLElementStyle]

The HTML `<style>` element indicates style information for a document. HTML documents can contain multiple `<style>` elements. See CSS Tutorial for further details.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`media`	Specifies media / device linked document is optimised for	Here
`scoped`	Indicates styles only apply to the element's parent element and that element's child elements	Here
`type`	Type of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object (typically the style property of an element) supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports the following additional properties, see also CSS Properties:

DOM Property Name	Corresponding CSS Property Name	More
`alignContent`	`align-content`	Here
`alignItems`	`align-items`	Here
`alignSelf`	`align-self`	Here
`animation`	`animation`	Here

animationDelay	animation-delay	Here
animationDirection	animation-direction	Here
animationDuration	animation-duration	Here
animationFillMode	animation-fill-mode	Here
animationIterationCount	animation-iteration-count	Here
animationName	animationName	Here
animationPlayState	animationPlayState	Here
animationTimingFunction	animationTimingFunction	Here
backfaceVisibility	backface-visibility	Here
background	background	Here
backgroundAttachment	background-attachment	Here
backgroundClip	background-clip	Here
backgroundColor	background-color	Here
backgroundImage	background-image	Here
backgroundOrigin	background-origin	Here
backgroundPosition	background-position	Here
backgroundRepeat	background-repeat	Here
backgroundSize	background-size	Here
border	border	Here
borderBottom	border-bottom	Here
borderBottomColor	border-bottom-color	Here
borderBottomLeftRadius	border-bottom-left-radius	Here
borderBottomRightRadius	border-bottom-right-radius	Here
borderBottomStyle	border-bottom-style	Here
borderBottomWidth	border-bottom-width	Here
borderCollapse	border-collapse	Here
borderColor	border-color	Here
borderImage	border-image	Here
borderImageOutset	border-image-outset	Here
borderImageRepeat	border-image-repeat	Here
borderImageSlice	border-image-slice	Here
borderImageSource	border-image-source	Here
borderImageWidth	border-image-width	Here
borderLeft	border-left	Here
borderLeftColor	border-left-color	Here
borderLeftStyle	border-left-style	Here
borderLeftWidth	border-left-width	Here
borderRadius	border-radius	Here
borderRight	border-right	Here
borderRightColor	border-right-color	Here
borderRightStyle	border-right-style	Here
borderRightWidth	border-right-width	Here
borderSpacing	border-spacing	Here
borderStyle	border-style	Here
borderTop	border-top	Here
borderTopColor	border-top-color	Here
borderTopLeftRadius	border-top-left-radius	Here
borderTopRightRadius	border-top-right-radius	Here
borderTopStyle	border-top-style	Here
borderTopWidth	border-top-width	Here

borderWidth	border-width	Here
bottom	bottom	Here
boxShadow	box-shadow	Here
boxSizing	box-sizing	Here
captionSide	caption-side	Here
clear	clear	Here
clip	clip	Here
color	color	Here
columnCount	column-count	Here
columnFill	column-fill	Here
columnGap	column-gap	Here
columnRule	column-rule	Here
columnRuleColor	column-rule-color	Here
columnRuleStyle	column-rule-style	Here
columnRuleWidth	column-rule-width	Here
columnSpan	column-span	Here
columnWidth	column-width	Here
columns	columns	Here
content	content	Here
counterIncrement	counter-increment	Here
counterReset	counter-reset	Here
cursor	cursor	Here
direction	direction	Here
display	display	Here
emptyCells	empty-cells	Here
filter	filter	Here
flex	flex	Here
flexBasis	flex-basis	Here
flexDirection	flex-direction	Here
flexFlow	flex-flow	Here
flexGrow	flex-grow	Here
flexShrink	flex-shrink	Here
flexWrap	flex-wrap	Here
cssFloat	float	Here
font	font	Here
fontFamily	font-family	Here
fontSize	font-size	Here
fontSizeAdjust	font-size-adjust	Here
fontStretch	font-stretch	Here
fontStyle	font-style	Here
fontVariant	font-variant	Here
fontWeight	font-weight	Here
hangingPunctuation	hanging-punctuation	Here
height	height	Here
justifyContent	justify-content	Here
left	left	Here
letterSpacing	letter-spacing	Here
lineHeight	line-height	Here
listStyle	list-style	Here
listStyleImage	list-style-image	Here

listStylePosition	list-style-position	Here
listStyleType	list-style-type	Here
margin	margin	Here
marginBottom	margin-bottom	Here
marginLeft	margin-left	Here
marginRight	margin-right	Here
marginTop	margin-top	Here
maxHeight	max-height	Here
maxWidth	max-width	Here
minHeight	min-height	Here
minWidth	min-width	Here
navDown	nav-down	Here
navIndex	nav-index	Here
navLeft	nav-left	Here
navRight	nav-right	Here
navUp	nav-up	Here
opacity	opacity	Here
order	order	Here
orphans	orphans	Here
outline	outline	Here
outlineColor	outline-color	Here
outlineOffset	outline-offset	Here
outlineStyle	outline-style	Here
outlineWidth	outline-width	Here
overflow	overflow	Here
overflowX	overflow-x	Here
overflowY	overflow-y	Here
Padding	padding	Here
paddingBottom	padding-bottom	Here
paddingLeft	padding-left	Here
paddingRight	padding-right	Here
paddingTop	padding-top	Here
pageBreakAfter	page-break-after	Here
pageBreakBefore	page-break-before	Here
pageBreakInside	page-break-inside	Here
perspective	perspective	Here
perspectiveOrigin	perspective-origin	Here
position	position	Here
quotes	quotes	Here
resize	resize	Here
right	right	Here
tabSize	tab-size	Here
tableLayout	table-layout	Here
textAlign	text-align	Here
textAlignLast	text-align-last	Here
textDecoration	text-decoration	Here
textDecorationColor	text-decoration-color	Here
textDecorationLine	text-decoration-line	Here
textDecorationStyle	text-decoration-style	Here
textIndent	text-indent	Here

textJustify	text-justify	Here
textOverflow	text-overflow	Here
textShadow	text-shadow	Here
textTransform	text-transform	Here
top	top	Here
transform	transform	Here
transformOrigin	transform-origin	Here
transformStyle	transform-style	Here
transition	transition	Here
transitionDelay	transition-delay	Here
transitionDuration	transition-duration	Here
transitionProperty	transition-property	Here
transitionTimingFunction	transition-timing-function	Here
unicodeBidi	unicode-bidi	Here
userSelect	user-select	Here
verticalAlign	vertical-align	Here
visibility	visibility	Here
whiteSpace	white-space	Here
widows	widows	Here
width	width	Here
wordBreak	word-break	Here
wordSpacing	word-spacing	Here
wordWrap	word-wrap	Here
zIndex	z-index	Here

The default style applicable to this element is shown here.

<summary>

[HTMLElementSummary]

The HTML `<summary>` element indicates a heading for a <details> element. It is new in HTML 5.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<sup>

[HTMLElementSup]

The HTML `<sup>` element indicates superscripted text.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<table>

[HTMLElementTable]

The HTML `<table>` element indicates a table. It typically includes one or more <tr> elements and, within them, <td> and/or <th> elements. More complicated table layouts can also include <caption>, <col>, <colgroup>, <tbody>, <tfoot> and <thead> elements.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align`, `bgcolor`, `border`, `cellpadding`, `cellspacing`, `frame`, `rules`, `summary` and `width` attributes, but these are no longer supported by HTML 5. Original draft versions of HTML 5 also included a `sortable` attribute, but this appears to have been dropped in later specifications and not to have been implemented so far by major browsers.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. It also supports the following additional properties and methods:

Additional properties:

Property	Description	More
`caption`	Returns the <caption> element of the table	Here
`rows`	Returns a collection of the <tr> elements of the table	Here
`tBodies`	Returns a collection of <tbody> elements of the table	Here
`tFoot`	Returns the <tfoot> element of the table	Here
`tHead`	Returns the <thead> element of the table	Here

Additional methods:

Method	Description	More
`createCaption()`	Creates empty <caption> element and adds to table	Here
`createTFoot()`	Creates empty <tfoot> element and adds to table	Here
`createTHead()`	Creates empty <thead> element and adds to table	Here
`deleteCaption()`	Removes first <caption> element from table	Here
`deleteRow()`	Removes a <tr> element from table	Here
`deleteTFoot()`	Removes first <tfoot> element from table	Here

`deleteTHead()`	Removes <thead> element from table	Here
`insertRow()`	Creates empty <tr> element and adds to table	Here

The default style applicable to this element is shown here.

<tbody>

[HTMLElementTbody]

The HTML `<tbody>` element indicates the body of a table. It appears inside a <table> element and is used in conjunction with <tfoot> and <thead> elements to differentiate between different parts of the table. This can allow browsers to scroll the table body independently of the header and footer, or to allow printing of the header and footer at the top and bottom of each page. A `<tbody>` element needs to come after any <caption>, <colgroup> and <thead> elements. It needs to contain one or more <tr> elements.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align`, `char`, `charoff` and `valign` attributes, but these are no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<td>

[HTMLElementTd]

The HTML `<td>` element indicates a table cell (within a table row). They appear inside <tr> elements. HTML tables contain two types of cells, i.e. header cells (<th> elements) and standard cells (`<td>` elements), and the two are by default formatted differently.

The attributes it can take (in addition to HTML global attributes and HTML event attributes) are:

Attribute	Description	More
`colspan`	Number of columns a table cell should span	Here
`headers`	One or more header cells a cell is related to	Here
`rowspan`	Number of rows a table cell should span	Here

It used to support the `abbr`, `align`, `axis`, `bgcolor`, `char`, `charoff`, `height`, `nowrap`, `scope`, `valign` and `width` attributes, but these are no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<textarea>

[HTMLElementTextarea]

The HTML `<textarea>` element indicates a multiline input control. It can hold an unlimited number of characters, and the text used is typically rendered in a fixed-width font. The size of the text area can be specified using the element's `cols` and `rows` attributes or using corresponding CSS attributes.

The attributes it can take (other than HTML global attributes and HTML event attributes) include:

Attribute	Description	More
`autofocus`	Specifies whether element should automatically get focus when page loads	Here
`cols`	Width of text area (in characters)	Here
`dirname`	Specifies text direction will be submitted	Here
`disabled`	Specified element(s) to be disabled	Here
`form`	Name of the form that element belongs to	Here
`maxlength`	Maximum number of characters allowed in an element	Here
`name`	Name of element	Here
`placeholder`	Short hint describing expected value of element	Here
`readonly`	Whether element is read-only	Here
`required`	Whether the element must be filled out before submitting form	Here
`rows`	Visible number of lines in a <textarea> element	Here
`wrap`	How text in a <textarea> element is to be wrapped when submitted in a form	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above (with the `maxlength` and `readonly` properties of the underlying element corresponding to the `maxLength` and `readOnly` properties of the DOM object). It also supports the following additional properties and methods:

Additional properties:

Property	Description	More
`defaultValue`	Sets / returns default value of element	Here
`type`	Returns type of form that contains element	Here
`value`	Sets / returns contents of element	Here

Additional methods:

Method	Description	More
`select()`	Selects entire contents of text area	Here

The default style applicable to this element is shown here.

<tfoot>

[HTMLElementTfoot]

The HTML <tfoot> element indicates the footer content in a table. It appears inside a <table> element and is used in conjunction with <tbody> and <thead> elements to differentiate between different parts of the table. This can allow browsers to scroll the table body independently of the header and footer, or to allow printing of the header and footer at the top and bottom of each page. A <tfoot> element needs to come after any <caption>, <colgroup> and <thead> elements and before any <tbody> and <tr> elements.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align`, `char`, `charoff` and `valign` attributes, but these are no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<th>

[HTMLElementTh]

The HTML <th> element indicates a table header cell (within a table row). They appear inside <tr> elements. HTML tables contain two types of cells, i.e. header cells (<th> element) and standard cells (<td> elements), and the two are by default formatted differently.

The attributes it can take (in addition to HTML global attributes and HTML event attributes) are:

Attribute	Description	More
`colspan`	Number of columns a table cell should span	Here
`headers`	One or more header cells a cell is related to	Here
`rowspan`	Number of rows a table cell should span	Here

It used to support the `abbr`, `align`, `axis`, `bgcolor`, `char`, `charoff`, `height`, `nowrap`, `scope`, `valign` and `width` attributes, but these are no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<thead>

[HTMLElementThead]

The HTML <thead> element indicates the header content of a table. It appears inside a <table> element and is used in conjunction with <tbody> and <tfoot> elements to differentiate between different parts of the table. This can allow browsers to scroll the table body independently of the header and footer, or to allow printing of the header and footer at the top and bottom of each page. A <thead> element needs to come after any <caption> and <colgroup> elements and before any <tbody>, <tfoot> and <tr> elements.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align`, `char`, `charoff` and `valign` attributes, but these are no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<time>

[HTMLElementTime]

The HTML `<time>` element indicates a (human-readable) date / time. It can be used to encode dates and times in a machine-readable fashion.

The attributes it can take (in addition to HTML global attributes and HTML event attributes) are:

Attribute	Description	More
`datetime`	Date and time of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. The default style applicable to this element is shown here.

<title>

[HTMLElementTitle]

The HTML `<title>` element indicates the title for the document. It appears in the <head> part of the document. It typically identifies the page title that appears in a browser toolbar, the page title that is by default added to a user's list of favourite pages within a browser and usually is the title shown for the page in search engine results.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. It also supports the following additional properties:

Property	Description	More
`text`	Sets / returns text of document title	Here

The default style applicable to this element is shown here.

<tr>

[HTMLElementTr]

The HTML `<tr>` element indicates a table row (within a table). It appears inside a <table> element and contains <td> and <th> elements representing individual cells within the table row.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the `align`, `bgcolor`, `char`, `charoff` and `valign` attributes, but these are no longer supported in HTML5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. It also supports the following additional properties and methods:

Additional properties:

Property	Description	More
`cells`	Returns collection of all <td> and <th> elements in row	Here
`rowIndex`	Returns position of row in rows collection of a <table> element	Here
`sectionRowIndex`	Returns position of row in rows collection of a <tbody>, <tfoot> or a <thead>	Here

Additional methods:

Method	Description	More
`deleteCell()`	Deletes a cell from table row	Here
`insertCell()`	Inserts a cell into table row	Here

The default style applicable to this element is shown here.

<track>

[HTMLElementTrack]

The HTML `<track>` element indicates a text track for a media element, i.e. a <video> or <audio>. It is new in HTML 5. It is used to specify subtitles, captions or other files containing text that should be visible when the media is playing.

The attributes it can take (in addition to HTML global attributes and HTML event attributes) are:

Attribute	Description	More
`default`	Default track / command to be enabled unless user preferences specify otherwise	Here
`kind`	Kind of text track	Here
`label`	Title / label of track or command or group of commands	Here
`src`	URL of track file	Here
`srclang`	Language of track text data	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports the following additional properties and methods:

Property	Description	More
readyState	Returns current state of track resource	Here
track	Returns TextTrack object representing the text track data of the track element	Here

The default style applicable to this element is shown here.

<tt>

[HTMLElementTt]

The HTML <tt> element indicates teletype text. It is not supported in HTML 5. Instead, use CSS.

<u>

[HTMLElementU]

The HTML <u> element indicates text that should be stylistically different from normal text. It is commonly used for underlining, even though the HTML 5 specification reminds developers that there are almost always other more appropriate ways of achieving a similar effect.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

[HTMLElementUl]

The HTML <ul> element indicates an unordered list. Inside the <ul> element should be one or more <li> elements identifying each entry in the unordered list.

The attributes it can take are HTML global attributes and HTML event attributes.

It used to support the compact and type attributes, but these are no longer supported by HTML 5.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<var>

[HTMLElementVar]

The HTML <var> element is a phrase element indicating a variable in computer code. It is not depreciated, but typically a richer effect can be achieved using CSS.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.

<video>

[HTMLElementVideo]

The HTML `<video>` element indicates a video or movie. It is new in HTML 5. Currently there are 3 supported video formats across most browsers: MP4 (i.e. MPEG 4 files with H264 video codec and AAC audio codec, MIME-type is video/mp4), WebM (i.e. WebM files with V8 video codec and Vorbis audio codec, MIME-type is video/webm) and Ogg (Ogg files with Theora video codec and Vorbis audio codec, MIME-type is video/ogg).

If the browser does not support `<video>` elements then any text between the `<video>` and `</video>` tags will be displayed.

The attributes it can take (in addition to HTML global attributes and HTML event attributes) are:

Attribute	Description	More
`autoplay`	Specifies whether media should start playing as soon as ready	Here
`controls`	Whether controls (such as play and pause buttons) should be displayed	Here
`height`	Height of element	Here
`loop`	Media to start over again when it finishes	Here
`muted`	Audio output should be muted	Here
`poster`	Image to be shown while video is downloading (or until user hits play)	Here
`preload`	If / how author thinks media should be loaded when page loads	Here
`src`	URL of media	Here
`width`	Width of element	Here

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods, and additional properties with the same name and meaning as the attributes of the underlying HTML element referred to above. It also supports DOM generic media properties and methods and the following additional properties and methods.

Additional properties:

Property	Description	More
`videoTracks`	Returns VideoTrackList object indicating available video tracks	Here

The default style applicable to this element is shown here.

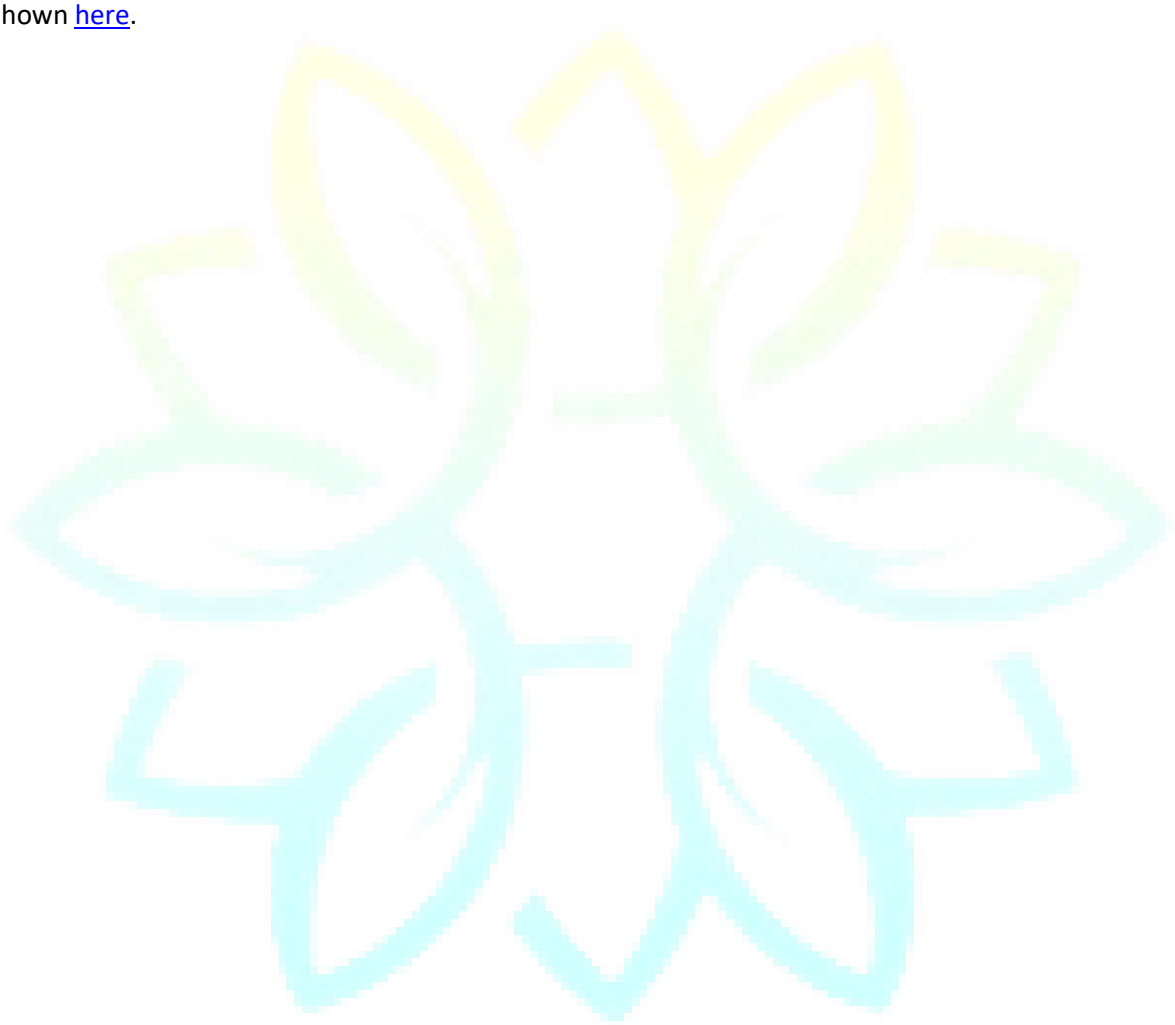
<wbr>

[HTMLElementWbr]

The HTML <wbr> element indicates a possible line-break (i.e. 'word break opportunity'). It is new in HTML 5. When a word is long and the browser might break text lines in wrong places then <wbr> elements offer scope to add word break opportunities.

The attributes it can take are HTML global attributes and HTML event attributes.

To create or access such an element in JavaScript see here. The corresponding HTML DOM object supports standard DOM properties and methods. The default style applicable to this element is shown here.



Appendix B: HTML Attributes

[HTMLAttributes]

Different HTML elements can have attributes that specify how they should be formatted or interpreted or allow further characterisation of the element. Attributes come in two basic types:

(a) Standard attributes, which describe or characterise the element further, and
(b) Event attributes, almost all of which begin with `on`…. These indicate what scripts should be run if an *event* occurs (e.g. the mouse button is clicked, an element is dragged, dropped or copied, etc.)

HTML attributes can also be set programmatically using JavaScript by modifying the properties of the corresponding HTML DOM elements. Listed here are the JavaScript DOM properties that correspond to most of the HTML attributes recognised in HTML 5.

Standard attributes

[HTMLStandardAttributes]

Different HTML elements can have attributes that specify how they should be formatted or interpreted or allow further characterisation of the element, which can be either standard attributes or event attributes. Standard attributes describe or characterise the element further and include:

Attribute	Description	Applicable to	More
`accept`	Specifies types of file accepted by server	<input>	Here
`accept-charset`	Specifies character encodings used for form submission	<form>	Here
`accesskey`	Shortcut key to activate/focus element	All	Here
`action`	Where to send form-data when form submitted	<form>	Here
`align`	Alignment versus surrounding elements		Here. Not supported in HTML 5 (instead use CSS)
`alt`	Specifies alternative text to show when original content fails to display	Various	Here
`async`	Indicates if script to be executed asyncronously	<script>	Here. Only for external scripts
`autocomplete`	Specifies whether element has autocomplete enabled	<form>, <input>	Here
`autofocus`	Specifies whether element should automatically get focus when page loads	Various	Here
`autoplay`	Specifies whether audio/video should start playing as soon as ready	<audio>, <video>	Here

bgcolor	Specifies background colour		Here. Not supported in HTML 5 (instead use CSS)
border	Specifies width of border		Here. Not supported in HTML 5 (instead use CSS)
challenge	Indicates value of element should be challenged when submitted	<keygen>	Here
charset	Specifies character encoding	<meta>, <script>	Here
checked	Specifies that the element should be pre-selected	<input>, <menuitem>	Here
cite	URL which explains the quote / deleted / inserted text	<blockquote>, <del>, <ins>, <q>	Here
class	One or more class names for the element	All	Here. Refers to a class in a CSS style
color	Text colour of element	All text elements	Here. Not supported in HTML 5 (instead use CSS)
cols	Width of text area (in characters)	<textarea>	
colspan	Number of columns a table cell should span	<td>, <th>	Here
content	Value associated with the http-equiv or name attribute	<meta>	Here
contenteditable	Indicates whether content is editable	All	Here
contextmenu	Specifies context menu (i.e. what appears when right-click)	All	Here
controls	Whether <audio> and <video> controls (such as play and pause buttons) should be displayed	<audio>, <video>	Here
coords	Specifies the coordinates of an <area>	<area>	Here
crossorigin	Specifies how element handles cross-origin requests	<img>, <link>	Here
data	URL of resource to be used by object	<object>	Here
data-*	Custom data private to page of application	All	Here
datetime	Date and time of element	<del>, <ins>, <time>	Here
default	Default track / command to be enabled unless user preferences specify otherwise	<menuitem>, <track>	Here
defer	Script to be executed only	<script>	Here. Only for

	when page has finished parsing		external scripts
`dir`	Text direction for element content	All	Here
`dirname`	Specifies text direction will be submitted	<input>, <textarea>	Here
`disabled`	Specified element(s) to be disabled	Various	Here
`download`	Target will be downloaded when user clicks on hyperlink	<a>, <area>	Here
`draggable`	Whether element is draggable	All	Here
`dropzone`	Whether dragged data is copied, moved or linked when dropped	All	Here
`enctype`	How form-data to be encoded when submitted to server	<form>	Here. Only for method = `post`
`for`	Specifies which form element(s) a label calculation is bound to	<label>, <output>	Here
`form`	Name of the form that element belongs to	Various	Here
`formaction`	Where to send form-data to when form submitted	<button>, <input>	Here. Only for type = `submit`
`formenctype`	How form-data should be encoded before sending it to a server	<button>, <input>	Here. Only for type = `submit`
`formmethod`	How to send form-data (i.e. which HTTP method to use)	<button>, <input>	Here. Only for type = `submit`
`formnovalidate`	Specifies that form-data should not be validated on submission	<button>, <input>	Here. Only for type = `submit`
`formtarget`	Specifies where to display the response that is received after submitting form	<button>, <input>	Here. Only for type = `submit`
`headers`	One or more header cells a cell is related to	<td>, <th>	Here
`height`	Height of element	Various (not all)	Here
`hidden`	Whether element is not relevant	All	Here
`high`	Value above which is considered a high value	<meter>	Here
`href`	URL of page the link goes to	<a>, <area>, <base>, <link>	Here
`hreflang`	Language of linked document	<a>, <area>, <link>	Here
`http-equiv`	HTTP header for information/value of attribute	<meta>	Here

icon	Icon for a command / menu item	<menuitem>	Here
id	Unique id for an element (for e.g. JavaScript)	All	Here
ismap	Image is a server-side image-map	<img>	Here
keytype	Specifies security algorithm of key	<keygen>	Here
kind	Kind of text track	<track>	Here
label	Title / label of track or command or group of commands	<menu>, <menuitem>, <option>, <optgroup>, <track>	Here
lang	Language of an element's content	All	Here
list	Refers to <datalist> that contains pre-defined options	<input>	Here
loop	Audio / video to start over again when it finishes	<audio>, <video>	Here
low	Value below which is considered a low value	<meter>	Here
manifest	Specifies address of document's cache manifest (for offline browsing)	<html>	Here
max	Maximum value	<input>, <meter>, <progress>	Here
maxlength	Maximum number of characters allowed in an element	<input>, <textarea>	Here
media	Specifies media / device linked document is optimised for	<a>, <area>, <link>, <source>, <style>	Here
method	Specifies HTTP method used when sending form-data	<form>	Here
min	Minimum value	<input>, <meter>	Here
multiple	Indicates that a user can enter more than one value	<input>, <select>	Here
muted	Audio output should be muted	<audio>, <video>	Here
name	Name of element (or of a piece of metadata for a <meta> element)	Various	Here
novalidate	Form should not be validated when submitted	<form>	Here
open	Whether details should be visible (i.e. open) to user	<details>	Here
optimum	Value deemed optimal for gauge	<meter>	Here
pattern	Regular expression that value of element is checked	<input>	Here

	against		
placeholder	Short hint describing expected value of element	<input>, <textarea>	Here
poster	Image to be shown while video is downloading (or until user hits play)	<video>	Here
preload	If / how author thinks audio / video should be loaded when page loads	<audio>, <video>	Here
radiogroup	Name of group of commands when menu item toggled	<menuitem>	Here
readonly	Whether element is read-only	<input>, <textarea>	Here
rel	Relationship between current document and linked document	<a>, <area>, <link>	Here
required	Whether the element must be filled out before submitting form	<input>, <select>, <textarea>	Here
reversed	List order should be descending (3, 2, 1) not ascending (1, 2, 3)		Here
rows	Visible number of lines in a <textarea> element	<textarea>	Here
rowspan	Number of rows a table cell should span	<td>, <th>	Here
sandbox	Allows an extra set of restrictions for the content of an <iframe> element	<iframe>	Here
scope	Indicates whether a header cell is a header for a column, row or groups of these	<th>	Here
scoped	Indicates styles only apply to the element's parent element and that element's child elements	<style>	Here
selected	Indicates that an <option> element should be pre-selected when the page loads	<option>	Here
shape	Specifies shape of an <area> element	<area>	Here
size	Specifies width in characters for <input> or number of visible options for <select>	<input>, <select>	Here
sizes	Specifies size of linked resource	<img>, <link>, <source>	Here
span	Number of columns to span	<col>, <colgroup>	Here
spellcheck	Indicates whether element is to have its spelling and	All	Here

	grammar checked		
src	URL of resource	Various	Here
srcdoc	HTML content of page to show in an <iframe>	<iframe>	Here
srclang	Language of track text data	<track>	Here. New in HTML 5, for kind = subtitles
srcset	URL of image to use in different situations	<img>, <source>	Here
start	Start value of an ordered list		Here
step	Accepted number intervals for an <input> element	<input>	Here
style	Inline CSS style for an element	All	Here
tabindex	Tab order of an element	All	Here
target	Specifies where / how to open the linked document (or where to submit the form)	<a>, <area>, <base>, <form>	Here
title	Extra information about element	All	Here
translate	Whether content of an element should be translated	All	Here
type	Type of element	Various	Here
usemap	Specifies an image as a client-side image-map	<img>, <object>	Here
value	Value of element	Various	Here
width	Width of element	Various (not all)	Here
wrap	How text in a <textarea> element is to be wrapped when submitted in a form	<textarea>	Here
xmlns	XML namespace attribute applicable to the webpage (if it needs to conform to XHTML)	<html>	Here

Some standard attributes can apply to essentially all HTML elements. These are called global attributes.

Event attributes

[HTMLEventAttributes]

Different HTML elements can have attributes that specify how they should be formatted or interpreted or allow further characterisation of the element, which can be either standard attributes or event attributes. Event attributes indicate what scripts should be run if an event occurs (e.g. the mouse button is clicked, an element is dragged, dropped or copied, etc). HTML5 added many more possible event attributes that can be assigned to HTML elements. In each case the value of the attribute is a script to be run when an event occurs.

HTML event attributes include:

Event Attribute	Description of event	Applicable to	More
`animationend`	When CSS animation ends	Any element with a CSS animation	Here
`animationiteration`	When CSS animation is repeated	Any element with a CSS animation	Here
`animationstart`	When CSS animation starts	Any element with a CSS animation	Here
`onabort`	If document or media loading is aborted	<audio>, <embed>, <input> (if type = `image`), <img>, <object>, <video>	Here
`onafterprint`	After document printed	<body>	Here
`onbeforeprint`	Before document printed	<body>	Here
`onbeforeunload`	Just before document unloaded	<body>	Here
`onblur`	When element loses focus	All visible elements	Here
`oncanplay`	When file ready to start playing (i.e. when buffered enough to begin)	<audio>, <embed>, <object>, <video>	Here
`oncanplaythrough`	When file ready to be played all way to end without pausing for buffering	<audio>, <video>	Here
`onchange`	When element value changed	All visible elements	Here
`onclick`	When element clicked	All visible elements	Here
`oncontextmenu`	When context menu triggered	All visible elements	Here
`oncopy`	When content of element being copied	All visible elements	Here
`oncuechange`	When cue changes	<track>	Here
`oncut`	When content of element being cut	All visible elements	Here
`ondblclick`	When element is double-clicked	All visible elements	Here
`ondrag`	When element being dragged	All visible elements	Here
`ondragend`	At end of drag operation	All visible elements	Here
`ondragenter`	When element has been dragged to a valid drop target	All visible elements	Here
`ondragleave`	When element leaves a valid drop target	All visible elements	Here
`ondragover`	When element being dragged over a valid drop target	All visible elements	Here
`ondragstart`	At start of drag operation	All visible elements	Here

ondrop	When dragged element is being dropped	All visible elements	Here
ondurationchange	When length of media changes	<audio>, <video>	Here
onemptied	When media unexpectedly becomes unavailable	<audio>, <video>	Here
onended	When media has reached end	<audio>, <video>	Here
onerror	When an error occurs	<audio>, <body>, <embed>, <img>, <object>, <script>, <style>, <video>, EventSource objects	Here
onfocus	When element gets focus	All visible elements	Here
onfocusin	When element is about to get focus (similar to onfocus except also bubbles)	All visible elements	Here
onfocusout	When element is about to lose focus (similar to onblur except also bubbles)	All visible elements	Here
onhashchange	When there has been changes to the anchor part of the URL	<body>	Here
oninput	When element gets user input	All visible elements	Here
oninvalid	When element is invalid	All visible elements	Here
onkeydown	When user is pressing key	All visible elements	Here
onkeypress	When user presses a key	All visible elements	Here
onkeyup	When user releases a key	All visible elements	Here
onload	When element finishes loading	<body>, <iframe>, <img>, <input>, <link>, <script>, <style>	Here
onloadeddata	When media data is loaded	<audio>, <video>	Here
onloadedmetadata	When metadata (dimensions, duration, …) loaded	<audio>, <video>	Here
onloadstart	Just before loading starts	<audio>, <video>	Here
onmessage	When message is triggered	For handling errors	Here
onmousedown	When mouse button is pressed down on an element	All visible elements	Here
onmouseenter	When mouse pointer moves over an element	All visible elements	Here
onmouseleave	When mouse pointer moves out of an element	All visible elements	Here
onmousemove	For as long as mouse pointer is moving over an element	All visible elements	Here

onmouseout	When mouse pointer moves out of an element	All visible elements	Here
onmouseover	When mouse pointer moves over an element	All visible elements	Here
onmouseup	When mouse button is released over an element	All visible elements	Here
onmousewheel	When mouse wheel is being scrolled over an element (depreciated: use onwheel instead)	All visible elements	Here
onoffline	When browser starts to work offline	<body>	Here
ononline	When browser starts to work online	<body>	Here
onopen	When a connection to an event source is opened	An EventSource object	Here
onpagehide	When user navigates away from a page	<body>	Here
onpageshow	When user navigates to a page	<body>	Here
onpaste	When user pastes content in an element	All visible elements	Here
onpause	When media is paused	<audio>, <video>	Here
onplay	When media is ready to start playing	<audio>, <video>	Here
onplaying	When media has actually started playing	<audio>, <video>	Here
onpopstate	When window's history changes	<body>	Here
onprogress	When browser is in process of getting media data	<audio>, <video>	Here
onratechange	When playback rate changes (e.g. user switches to fast forward)	<audio>, <video>	Here
onreset	When reset button in a form is clicked	<form>	Here
onresize	When browser window is being resized	<body>	Here
onscroll	When element's scrollbar is being scrolled	All visible elements	Here
onsearch	When user writes something in search field (for an <input> element of type = search)	<input>	Here
onseeked	When seeking attribute is set to false (i.e. seeking finished)	<audio>, <video>	Here
onseeking	When seeking attribute is set to true (i.e. seeking is active)	<audio>, <video>	Here
onselect	When element gets selected	All visible elements	Here
onshow	When <menu> element is	<menu>	Here

	shown as a context menu		
onstalled	When browser is unable to fetch media data (for whatever reason)	<audio>, <video>	Here
onstorage	When web storage area is updated	<body>	Here
onsubmit	When a form is submitted	<form>	Here
onsuspend	When fetching media data is stopped before completely loaded (for whatever reason)	<audio>, <video>	Here
ontimeupdate	When playing position has changed (e.g. user fast forwards to new position in media)	<audio>, <video>	Here
ontoggle	When user opens or closes <details> element	<details>	Here
ontouchcancel	When touch is interrupted	Touch-sensitive elements	Here
ontouchend	When finger is removed from touch screen	Touch-sensitive elements	Here
ontouchmove	When finger is dragged across touch screen	Touch-sensitive elements	Here
ontouchstart	When finger is placed on touch screen	Touch-sensitive elements	Here
onunload	When page has unloaded (or browser window closed)	<body>	Here
onvolumechange	When volume changed (or muted)	<audio>, <video>	Here
onwaiting	When media has paused but is expected to resume (e.g. media has paused to buffer more data)	<audio>, <video>	Here
onwheel	When mouse wheel rolls up or down over an element	All visible elements	Here
transitionend	When CSS transition ends	Any element with a CSS transition	Here

Most of these event attributes are new in HTML 5. The ones that are not are: onabort, onblur, onchange, onclick, oncopy, oncut, ondblclick, onfocus, onkeydown, onkeypress, onkeyup, onload, onmousedown, onmousemove, onmouseout, onmouseover, onmouseup, onmousewheel, onpaste, onsearch, onselect, onsubmit and onunload.

Global attributes

[HTMLGlobalAttributes]

Different HTML elements can have attributes that specify how they should be formatted or interpreted or allow further characterisation of the element, which can be either standard attributes or event attributes. Some standard attributes can apply to essentially all HTML elements. These 'global' attributes include:

Attribute	Description	Applicable to	More
`accesskey`	Shortcut key to activate/focus element	All	Here
`class`	One or more class names for the element	All	Here. Refers to a class in a CSS style
`contenteditable`	Indicates whether content is editable	All	Here. New in HTML 5
`contextmenu`	Specifies context menu (i.e. what appears when right-click)	All	Here. New in HTML 5
`data-*`	Custom data private to page of application	All	Here. New in HTML 5
`dir`	Text direction for element content	All	Here. New in HTML 5
`draggable`	Whether element is draggable	All	Here. New in HTML 5
`dropzone`	Whether dragged data is copied, moved or linked when dropped	All	Here. New in HTML 5
`hidden`	Whether element is not relevant	All	Here. New in HTML 5
`id`	Unique id for an element (for e.g. JavaScript)	All	Here
`lang`	Language of an element's content	All	Here
`spellcheck`	Indicates whether element is to have its spelling and grammar checked	All	Here. New in HTML 5
`style`	Inline CSS style for an element	All	Here
`tabindex`	Tab order of an element	All	Here
`title`	Extra information about element	All	Here
`translate`	Whether content of an element should be translated	All	Here. New in HTML 5

Individual HTML Attributes:

accept

[HTMLAttributeAccept]

The HTML `accept` attribute specifies the types of file accepted by the server. It applies to <input> elements but only if type = `file` (and, prior to HTML 5, to <area> elements).

Valid attribute values (when used with <area> elements) include:

Value	Description

file_type	Not supported in HTML 5. Alternative text to display

Valid attribute values (when used with <input> elements) include:

Value	Description
file_extension	A file extension starting with a full stop, e.g. `.png`, `.jpg`, `.pdf`, `.doc`
media_type	A valid media type, see e.g. http://www.iana.org/assignments/media-types/media-types.xhtml
`audio/*`	Indicates any audio (sound) file is acceptable
`image/*`	Indicates any image file is acceptable
`video/*`	Indicates any video file is acceptable

More than one value can applied, if separated by commas.

accept-charset

[CNS website page: HTMLAttributeAcceptCharset, © CNS Global 2024]

The HTML `accept-charset` attribute specifies the character encodings used for submission of a <form> element.

Valid attribute values (when used with <form> elements) include:

Value	Description
character_set	Character encodings to be used when submitting the form

Common *character_sets* include:

- `UTF-8`: Unicode
- `ISO-8859-1`: character encoding for Latin alphabet

Multiple character_sets are acceptable and in HTML 5 need to be delimited (separated) by spaces. The default value is the reserved string `UNKNOWN`, which indicates that the encoding is the same as that for the document containing the <form> element.

accesskey

[HTMLAttributeAccesskey]

The HTML `accesskey` attribute indicates the shortcut key used to activate / focus an element. In HTML 5 it can in principle be used with any element, although in practice it may not be of much use with some elements. Different browsers use different ways of accessing shortcut keys (sometimes using the Alt key (or Alt and Shift keys simultaneously) or the Control key, in combination with a specified character.

Valid attribute values (when used with <form> elements) include:

Value	Description
character	The shortcut key character used to activate / focus the element

action

[HTMLAttributeAction]

The HTML `action` attribute indicates where to send form-data for a <form> element when the form is submitted.

Valid attribute values (when used with a <form> element) include:

Value	Description
URL	Where to send the form-data when form is submitted

align

[HTMLAttributeAlign]

The HTML `align` attribute indicates the alignment of the element versus surrounding elements. It is not supported in HTML 5 (instead use CSS, e.g. `<div style="text-align:center"> … </div>`).

alt

[HTMLAttributeAlt]

The HTML `alt` attribute indicates the alternative text to show when original content (e.g. an image) fails to display. It applies to <area>, <img> and <input> elements.

There are several possible reasons why an image might not display, e.g. there might be a slow connection, the content location might be wrongly specified or the user might be using a screen reader because he or she is partly sighted). Some old browsers showed the value of the `alt` attribute as a tooltip, but modern browsers use the title attribute instead for this purpose.

Valid attribute values (when used with <area>, <img> and <input> elements) include:

Value	Description
text	Alternative text to display

aminationend

[HTMLAttributeAnimationend]

The HTML `aminationend` attribute specifies the event that is triggered when a CSS animation ends. It applies to HTML elements that have CSS animatable elements. It seems to be necessary to set it using JavaScript.

aminationiteration

[HTMLAttributeAnimationiteration]

The HTML `aminationiteration` attribute specifies the event that is triggered when a CSS animation is repeated. It applies to HTML elements that have CSS animatable elements. It seems to be necessary to set it using JavaScript.

aminationstart

[HTMLAttributeAnimationstart]

The HTML `aminationstart` attribute specifies the event that is triggered when a CSS animation starts. It applies to HTML elements that have CSS animatable elements. It seems to be necessary to set it using JavaScript.

async

[HTMLAttributeAsync]

The HTML `async` attribute indicates if a script is to be executed asyncronously. It applies to <script> elements. It only in practice applies to external scripts, so should only be used if the `src` attribute is also present.

The `async` and defer attributes work in tandem as follows:

- If `async` is present then the (external) script is executed asynchronously with the rest of the page (with the script being executed while the page continues to be parsed)
- If `async` is not present but `defer` is present then the (external) script is executed when the page has finished parsing
- If neither `async` or `defer` is present then the (external) script is fetched and executed immediately, before further parsing of the page

Valid attribute values (when used with <script> elements) include:

Value	Description
`async`	Script should be executed asyncronously

autocomplete

[HTMLAttributeAutocomplete]

The HTML `autocomplete` attribute indicates whether an element has autocomplete capability enabled. This enables the browser to display options to fill in the field, based on previously typed characters. It applies to <form> and <input> elements (if the <input> element is type: `text`, `search`, `url`, `tel`, `email`, `password`, `datepickers`, `range` or `color`). Sometimes an autocomplete function needs to be enabled within the browser for autocomplete to work.

Valid attribute values (when used with <form> and <input> elements) include:

Value	Description
`on`	Form should have autocomplete on
`off`	Form should have autocomplete off

autofocus

[HTMLAttributeAutofocus]

The HTML `autofocus` attribute indicates whether an element should automatically get focus when the page loads. It applies to <button>, <input>, <keygen>, <select> and <textarea> elements.

Valid attribute values (when used with <button>, <input>, <keygen>, <select> and <textarea> elements) include:

Value	Description
`autofocus`	Element should automatically get focus when page loads

autoplay

[HTMLAttributeAutoplay]

The HTML `autoplay` attribute indicates whether an audio or video should start playing as soon as it is ready. It applies to <audio> and <video> elements.

Valid attribute values (when used with an <audio> and <video> element) include:

Value	Description
`autoplay`	Media to start playing as soon as ready

bgcolor

[HTMLAttributeBgcolor]

The HTML `bgcolor` attribute indicates the background colour of an element. It is no longer supported in HTML 5 (instead use CSS, e.g. `<div style="background-color:yellow">…</div>`).

border

[HTMLAttributeBorder]

The HTML `border` attribute indicates the width of the border of an element. It is no longer supported in HTML 5 (instead use CSS).

challenge

[HTMLAttributeChallenge]

The HTML `challenge` attribute indicates that the value of an element should be challenged when submitted. It applies to <keygen> elements. Note: it appears likely that <keygen> elements will be dropped from future versions of HTML so it may be desirable not to use <keygen> elements.

Valid attribute values (when used with <keygen> elements) include:

Value	Description
`challenge`	Value of element should be challenged when submitted

charset

[HTMLAttributeCharset]

The HTML `charset` attribute specifies the character encoding to use. It applies to <meta> and <script> elements.

Common values for this attribute include:

- UTF-8: the character encoding for Unicode
- ISO-8859-1: the character encoding for the Latin alphabet

It can be overridden for a specific element by setting the lang attribute of that element. The `charset` attribute is new in HTML 5 and replaces the need to set the content type via HTML such as: `<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">` (although using the http-equiv approach is still allowed).

Valid attribute values (when used with <meta> and <script> elements) include:

Value	Description
character_set	The character encoding for the document

checked

[HTMLAttributeChecked]

The HTML `checked` attribute specifies that the (sub)-element should be pre-selected (i.e. 'checked') when the page first loads. It applies to <input> elements (if type = `checkbox` or type = `radio`). It also applies to <menuitem> elements (but these are not currently supported by many browsers)

Valid attribute values (when used with an <input> element) include:

Value	Description
`checked`	Sub-element should be pre-selected

Valid attribute values (when used with a <menuitem> element) include:

Value	Description
`checked`	Indicates that command/menu item should be checked when page loads. Only applies to type = `radio` or type = `checkbox`

cite

[HTMLAttributeCite]

The HTML `cite` attribute provides a URL which explains a quote, deleted text or inserted text. It applies to <blockquote>, <del>, <ins> and <q> elements. It has no visual effect in typical web browsers but can be used by screen readers.

Valid attribute values (when used with <blockquote>, <del>, <ins> elements) include:

Value	Description
URL	Source of quote, deletion or insertion

class

[HTMLAttributeClass]

The HTML `class` attribute indicates one or more style class names (as per CSS) that apply to the element. If more than one class is to be applied to the same element then they should be separated by a space.

Valid attribute values include:

Value	Description
CSSclass	CSS style class

color

[HTMLAttributeColor]

The HTML `color` attribute indicates the colour of the text of an element. It is no longer supported in HTML 5 (instead use CSS).

cols

[HTMLAttributeCols]

The HTML `cols` attribute indicates the visible width of a <textarea> element (in number of characters).

Valid attribute values (when used with <textarea> elements) include:

Value	Description
integer	Visible width (in characters) of text area

The visible height of a <textarea> element can be set using the rows attribute. The size of a <textarea> element can also be set using CSS height and width properties.

colspan

[HTMLAttributeColspan]

The HTML `colspan` attribute indicates the number of columns a table cell should span. It applies to <td> and <th> elements.

A value of zero, i.e. using colspan="0", in theory has a special meaning, namely that the cell should be spanned to the last column of the column group, but this is not recognised by some browsers.

Valid attribute values (when used with <td> and <th> element) include:

Value	Description
integer	Number of columns a cell should span

content

[HTMLAttributeContent]

The HTML `content` attribute indicates the value associated with the http-equiv or name attribute within a <meta> element.

Valid attribute values (for <meta>) include:

Value	Description
text	Value associated with the relevant http-equiv or name attribute

contenteditable

[HTMLAttributeContenteditable]

The HTML `contenteditable` attribute indicates whether content of an element is editable.

Valid attribute values include:

Value	Description
`contenteditable`	Content of element is editable

contextmenu

[HTMLAttributeContextmenu]

The HTML `contextmenu` attribute indicates the context menu (i.e. what appears when the mouse is right-clicked). At the time of writing (early 2018) it did not appear to be supported by many browsers.

Valid attribute values include:

Value	Description
text	Text to display when mouse is right-clicked

controls

[HTMLAttributeControls]

The HTML `controls` attribute indicates whether <audio> and <video> controls (such as play and pause buttons) should be displayed.

Valid attribute values (when used with <audio> and <video> elements) include:

Value	Description
`controls`	Controls should be displayed (e.g. play / pause button, fast forward etc.

coords

[HTMLAttributeCoords]

The HTML `coords` attribute indicates the coordinates of an <area>. It, together with the shape attribute specify the size, shape and position of the area.

Valid attribute values (for <area>) include:

Value	Description
x1, *y1*, *x2*, *y2*	Coordinates of the left, top (*x1*, *y1*) and right, bottom (*x2*, *y2*) corners of a rectangle, if `shape = "rect"`
x, *y*, *r*	Coordinates of the circle centre (*x*, *y*) and circle radius (*r*), if `shape = "circle"`
x1, *y1*, *x2*, *y2*,..., *xn*, *yn*	Coordinates of corners of polygon. If the first and last coordinate pairs are not the same then the browser will add another coordinate pair to complete the polygon, if `shape = "poly"`

crossorigin

[HTMLAttributeCrossorigin]

The HTML `crossorigin` attribute indicates how the element handles cross-origin requests. It can apply to <img> and <link> elements (and some other elements for some browsers).

Valid attribute values (when used with <img> and <link> elements) include:

Value	Description
	(default), i.e. blank. CORS will not be used at all.
`anonymous`	CORS requests for element will not have credentials flag set, i.e. no user credentials will be exchanged via e.g. cookies, client-side SSL certificates or HTTP authentication
`use-credentials`	CORS requests for element will have credentials flag set, i.e. request will provide credentials

data

[HTMLAttributeData]

The HTML `data` attribute specifies the URL of a resource to be used by an <object> element.

Valid attribute values (when used with <object> elements) include:

Value	Description
URL	URL of resource used by element

data-*

[HTMLAttributeDataCustom]

The HTML `data-*` attribute provides a means of storing custom data specific to a page. It can be applied to all HTML elements and can be accessed by JavaScript embedded within the page.

The `data-*` attributes have two components:

(a) The name, which is the * part of the overall attribute name, which should not contain any uppercase letters
(b) The attribute value, which can be any string

E.g.

```
<ul>
  <li data-continent="Europe">Spain</li>
  <li data-continent="Asia">Japan</li>
</ul>
```

datetime

[HTMLAttributeDatetime]

The HTML `datetime` attribute specifies the date and time of a <del>, <ins> or <time> element.

Valid attribute values (when used with <del>, <ins> elements) include:

Value	Description
YYYY-MM-DDThh:mm:ssTZD	A date

Valid attribute values (when used with <time> elements) include:

Value	Description
datetime	A machine-readable date/time for the <time> element

default

[HTMLAttributeDefault]

The HTML `default` attribute specifies the default <menuitem> or <track> element that will be enabled unless user preferences specify otherwise.

Valid attribute values (when used with <menuitem> and <track> elements) include:

Value	Description
`default`	Marks relevant element as default

defer

[HTMLAttributeDefer]

The HTML `defer` attribute specifies that JavaScript within (or refenced by) a <script> should only be executed once the page has finished being parsed by the browser. It only in practice applies to external scripts.

The async and `defer` attributes work in tandem as follows:

- If `async` is present then the (external) script is executed asynchronously with the rest of the page (with the script being executed while the page continues to be parsed)
- If `async` is not present but `defer` is present then the (external) script is executed when the page has finished parsing
- If neither `async` or `defer` is present then the (external) script is fetched and executed immediately, before further parsing of the page

Valid attribute values (when used with <script> elements) include:

Value	Description
`defer`	Script should be deferred until page has finished being parsed by the browser

dir

[HTMLAttributeDir]

The HTML `dir` attribute specifies the direction of the text content of an element.

Valid attribute values (for <bdo>) include:

Value	Description
`ltr`	Left-to-right
`rtl`	Right-to-left

dirname

[HTMLAttributeDirname]

The HTML `dirname` attribute indicates that the text direction for the content of an element will be submitted. It applies to <input> and <textarea> elements. Currently, not all major browsers support this attribute.

The value of the `dirname` attribute is always the name of the input field, followed by `.dir`, i.e. valid attribute values (when used with <input> and <textarea> elements) are:

Value	Description

inputfieldname`.dir`	Indicates that the text direction of the input field with name *inputfieldname* will be specified

disabled

[HTMLAttributeDisabled]

The HTML `disabled` attribute indicates that the element or group of elements should be disabled. It applies to <button>, <fieldset>, <input>, <keygen>, <menuitem>, <optgroup>, <option>, <select> and <textarea> elements.

Valid attribute values (when used with <button>, <fieldset>, <input>, <keygen>, <menuitem>, <optgroup>, <option>, <select> and <textarea> elements) include:

Value	Description
`disabled`	Element (button, group of related form elements etc.) should be disabled

download

[HTMLAttributeDownload]

The HTML `download` attribute indicates that the target resource will be downloaded when the user clicks on the hyperlink. It applies to <a> and <area> elements.

Valid attribute values (when used with <a> and <area> elements) include:

Value	Description
filename	Resource to be downloaded

draggable

[HTMLAttributeDraggable]

The HTML `draggable` attribute indicates whether an element is draggable.

Valid attribute values include:

Value	Description
`draggable`	Element is draggable

dropzone

[HTMLAttributeDropzone]

The HTML `dropzone` attribute indicates whether dragged material is copied, moved or linked to when it is dropped. It does not seem currently to be supported by many browsers

Valid attribute values include:

Value	Description

`copy`	Dropping will create a copy of the element that was dragged
`move`	Dropping will result in the element that was dragged being moved to this new location
`link`	Dropping will create a link to the dragged data

enctype

[HTMLAttributeEnctype]

The HTML `enctype` attribute indicates how form-data should be encoded when submitted to the server. It applies to <form> elements and then only for method = `post`.

Valid attribute values (for <form>) include:

Value	Description
`application/x-www-form-urlencoded`	(Default). Input control names and values are URL encoded. Each name value pair is separated by a '`&`' and the name is separated from the value by '=' (as per a usual HTML query string)
`multipart/form-data`	Used for submitting forms that contain files, binary data and other non-ASCII data
`text/plain`	Obsolete. No encoding is applied (is only retained for old browser compatibility)

for

[HTMLAttributeFor]

The HTML `for` attribute indicates which form element(s) a label calculation is linked to. It applies to <label> and <output> elements.

Valid attribute values (when used with <label> elements) include:

Value	Description
elementID	Indicates to which form element the <label> belongs

Valid attribute values (when used with <output> elements) include:

Value	Description
elementID	Indicates relationship between calculation result and elements used calculation

form

[HTMLAttributeForm]

The HTML `form` attribute indicates the name of the form to which the element belongs. It applies to <button>, <fieldset>, <input>, <keygen>, <label>, <meter>, <object>, <output>, <select> and <textarea> elements.

Valid attribute values (when used with <button>, <fieldset>, <input>, <keygen>, <label>, <meter>, <object>, <output>, <select> and <textarea> elements) include:

Value	Description
formID	One or more forms to which element belongs

formaction

[HTMLAttributeFormaction]

The HTML `formaction` attribute indicates where to send form-data to when a form is submitted. It applies to <button> and <input> elements and then only for type = `submit`.

Valid attribute values (when used with <button> and <input> elements) include:

Value	Description
URL	Where form-data is sent to when a form is submitted. For <button> elements it only applies if the button type = "`submit`"

formenctype

[HTMLAttributeFormenctype]

The HTML `formenctype` attribute indicates how form-data should be encoded before sending it to a server. It applies to <button> and <input> elements and then only for type = `submit` or type = `image`. It overrides the enctype attribute of the <form> element containing the element.

Valid attribute values (when used with <button> and <input> elements include:

Value	Description
`application/x-www-form-urlencoded`	(Default). All characters are URL encoded before being sent (with spaces converted to + characters and special characters converted to ASCII Hex values
`multipart/form-data`	No characters encoded
`text/plain`	Spaces converted to + characters but no conversion applied to (other) special characters

formmethod

[HTMLAttributeFormmethod]

The HTML `formmethod` attribute indicates how to send form-data (i.e. which HTTP method to use). It applies to <button> and <input> elements and then only for type = `submit`.

Valid attribute values (when used with <button> and <input> elements) include:

Value	Description
HTTPmethod	HTTP method (get or post). See here for more details

formnovalidate

[HTMLAttributeFormnovalidate]

The HTML `formnovalidate` attribute indicates that form-data should not be validated prior to submission to server. It applies to <button> and <input> elements and then only for type = `submit`.

Valid attribute values (when used with <button> and <input> elements) include:

Value	Description
`formnovalidate`	Do not validate form

formtarget

[HTMLAttributeFormtarget]

The HTML `formtarget` attribute indicates where to display the response that is received after submitting form. It applies to <button> and <input> elements and then only for type = `submit`.

Valid attribute values (when used with <button> and <input> elements) include:

Value	Description
`_blank`	Opens linked document in a new window or tab
`_self`	Opens linked document in parent frame
`_parent`	(default value). Opens linked document in the same window or tab as was clicked
`_top`	Opens linked document in full body of the window
framename	Opens linked document in named frame

headers

[HTMLAttributeHeaders]

The HTML `headers` attribute identifies one or more header cells that a specific cell is related to. It applies to <td> and <th> elements.

Valid attribute values (when used with <td> and <th> element) include:

Value	Description
header_id	One or more header cells a cell is related to

height

[HTMLAttributeHeight]

The HTML `height` attribute indicates the height of an element. It applies to <canvas>, <embed>, <iframe>, <img>, <input>, <object> and <video> elements.

Valid attribute values (when used with <canvas>, <embed>, <iframe>, <img>, <input>, <object> and <video> elements) include:

Value	Description
number	Width of element or embedded content in pixels, e.g. `width="20"`
percentage	Width as a percentage of surrounding element, e.g. `width="30%"`

hidden

[HTMLAttributeHidden]

The HTML `hidden` attribute indicates whether an element is hidden.

Valid attribute values include:

Value	Description
`hidden`	Element is hidden

high

[HTMLAttributeHigh]

The HTML `high` attribute indicates a range that is considered to constitute a high value for a <meter> element.

Valid attribute values (when used with <meter> elements) include:

Value	Description
number	(Floating point) number defining number above which value is deemed 'high'. Should be lower than the max attribute and higher than the low attribute

href

[HTMLAttributeHref]

The HTML `href` attribute indicates the URL of the page that link goes to (or for the <base> element the URL that forms the base for relative URLS). It applies to <a>, <area>, <base> and <link> elements.

Valid attribute values (when used with <a>, <area>, <base>, <link> elements) include:

Value	Description
URL	URL location of linked document

If an element has an `href` attribute then the corresponding DOM object usually supports the following additional properties which can be thought of as variants of the `href` attribute:

Value	Description
`hash`	Anchor part of `href` attribute
`host`	Hostname and port part of `href` attribute
`hostname`	Hostname part of `href` attribute
`origin`	Returns protocol, hostname and port part of `href` attribute

password	Password part of href attribute
pathname	Pathname part of href attribute
port	Port part of href attribute
protocol	Protocol part of href attribute
search	Querystring part of href attribute
username	Username part of href attribute

hreflang

[HTMLAttributeHreflang]

The HTML hreflang attribute indicates the language of the linked document. It applies to <a>, <area> and <link> elements.

Valid attribute values (when used with <a>, <area>, <link> elements) include:

Value	Description
language-code	Language of text in linked document

http-equiv

[HTMLAttributeHttpEquiv]

The HTML http-equiv attribute provides the HTTP header for the information / value of an attribute within a <meta> element.

Valid attribute values (when used with <meta>) include:

Value	Description
content-type	The character encoding for the document, e.g.: <meta http-equiv="content-type" content="text/html; charset=UTF-8"> Note that using HTML 5 it is now possible to set the character set more directly, using e.g.: <meta charset="UTF-8">
default-style	The preferred style sheet to use for the page, e.g.: <meta http-equiv="default-style" content="*preferred stylesheet*"> The value of the relevant content attribute must either match the value of the title attribute of a link element (linked to an external style sheet) or a style element in the same document
refresh	The time interval for the document to refresh itself, e.g. <meta http-equiv="refresh" content="60"> It is recommended that this option is used sparingly, since it takes control of the page away from the user. Often better will be to achieve a similar effect using JavaScript

icon

[HTMLAttributeIcon]

The HTML `icon` attribute indicates the icon that should be used for a <menuitem> element.

Valid attribute values (when used with <menuitem> elements) include:

Value	Description
URL	Location of icon

id

[HTMLAttributeId]

The HTML `id` attribute indicates the unique id (identifier) for an element.

Valid attribute values include:

Value	Description
text	Unique id (identifier)

ismap

[HTMLAttributeIsmap]

The HTML `ismap` attribute indicates if an <img> element is a server-side image-map.

Valid attribute values (when used with <img> elements) include:

Value	Description
`ismap`	Specifies whether the <img> element is part of a server-side image-map, i.e. has clickable areas. The click coordinates are then sent to the server as part of the URL query string. It is only allowed if the <img> element is a descendant of an <a> element with a valid `href` attribute.

keytype

[HTMLAttributeKeytype]

The HTML `keytype` attribute specifies the security algorithm of a key. It applies to <keygen> elements.

Valid attribute values (when used with <keygen> elements) include:

Value	Description
`rsa`	(Default). Use RSA security algorithm (user may be given choice of key strength)
`dsa`	Use DSA security algorithm (user may be given choice of key strength)
`ec`	Use EC security algorithm (user may be given choice of key strength)

kind

[HTMLAttributeKind]

The HTML `kind` attribute specifies the kind of a text track (e.g. whether a subtitle). It applies to <track> elements.

Valid attribute values (when used with <track> elements) include:

Value	Description
`captions`	Track relates to translation of dialogue
`chapters`	Track relates to chapter titles (helps when navigating the media resource)
`descriptions`	Track relates to text description of video content
`metadata`	Track defines content used by scripts
`subtitles`	Track defines subtitles

label

[HTMLAttributeLabel]

The HTML `label` attribute specifies the title of the track or group (for <optgroup>, <option> and <track> elements) or the visible label to give to a menu (for <menu> elements).

Valid attribute values (when used with <menu>, <optgroup>, <option> and <track>elements) include:

Value	Description
text	Visible label

lang

[HTMLAttributeLang]

The HTML `lang` attribute specifies the language of an element's content.

Valid attribute values include:

Value	Description
language-code	Language of content

list

[HTMLAttributeList]

The HTML `list` attribute specifies the <datalist> element that contains pre-defined options for an <input> element.

Valid attribute values (when used with <input> elements) include:

Value	Description
datalist_id	ID of relevant <datalist> element

loop

[HTMLAttributeLoop]

The HTML `loop` attribute specifies whether an <audio> or <video> element is to start over again when it finishes.

Valid attribute values (when used with <audio> and <video> elements) include:

Value	Description
`loop`	Media to start over again each time it finishes

low

[HTMLAttributeLow]

The HTML `low` attribute indicates a range that is considered to constitute a low value for a <meter> element.

Valid attribute values (when used with <meter> elements) include:

Value	Description
number	(Floating point) number defining number below which value is deemed 'low'. Should be higher than the min attribute and lower than the high attribute

manifest

[HTMLAttributeManifest]

The HTML `manifest` attribute specifies the address of the document's cache manifest (for offline browsing). It applies to <html> elements.

Valid attribute values (when used with <html> elements) include:

Value	Description
URL	URL of document's cache manifest (facilitates offline browsing)

max

[HTMLAttributeMax]

The HTML `max` attribute specifies the maximum value applicable to an <input>, <meter> or <progress> element.

Valid attribute values (when used with <input>, <meter> elements) include:

Value	Description
number	Maximum (numerical) value for element
date	Maximum (date) value for an <input> element

maxlength

[HTMLAttributeMaxlength]

The HTML `maxlength` attribute specifies the maximum number of characters allowed in an <input> or <textarea> element.

Valid attribute values (when used with <input> and <textarea> elements) include:

Value	Description
integer	Maximum number of characters allowed to be inputted using element

media

[HTMLAttributeMedia]

The HTML `media` attribute specifies the media or device that the linked document is optimised for. It applies to <a>, <area>, <link>, <source> and <style> elements.

Valid attribute values (when used with <a>, <area>, <link>, <source> and <style> elements) include:

Value	Description
media-query	Indicates the media or device type the target URL is optimised for

method

[HTMLAttributeMethod]

The HTML `method` attribute specifies the HTTP method used when sending form-data. It applies to <form> elements.

Valid attribute values (for <form>) include:

Value	Description
HTTPmethod	HTTP method (get or post). See here for more details

min

[HTMLAttributeMin]

The HTML `min` attribute specifies the minimum value applicable to an <input> or <meter> element.

Valid attribute values (when used with <input>, <meter> elements) include:

Value	Description
number	Minimum (numerical) value for element
date	Minimum (date) value for an <input> element

multiple

[HTMLAttributeMultiple]

The HTML `multiple` attribute indicates that a user can enter more than one value into an <input> or <select> element.

Valid attribute values (when used with <input> and <select> elements) include:

Value	Description
`multiple`	Indicates that more than one value can be entered into element

muted

[HTMLAttributeMuted]

The HTML `muted` attribute indicates whether the audio output of an <audio> or <video> element should be muted.

Valid attribute values (when used with <audio> and <video> elements) include:

Value	Description
`muted`	Audio output should be muted

name

[HTMLAttributeName]

The HTML `name` attribute generally specifies the name of an element. It applies to <button>, <fieldset>, <form>, <iframe>, <input>, <keygen>, <map>, <meta>, <object>, <output>, <param>, <select> and <textarea> elements.

Valid attribute values (when used with <button>, <fieldset>, <form>, <iframe>, <input>, <keygen>, <object>, <output>, <select> and <textarea> elements) include:

Value	Description
name	Name for element or associated element

Valid attribute values (when used with <map> elements) include:

Value	Description
name	Name associated with the <img> element's usemap attribute that creates a relationship between the image and the map

Valid attribute values (when used with <meta> elements) include:

Value	Description
`application-name`	Name of web application to which page is associated
`author`	Author of document
`description`	Description of page (often picked up by search engines to show with

	results of searches)
`generator`	One or more software packages that have generated the document
`keywords`	A comma-separated list of keywords relevant to the page (again helpful for search engines). Specifying this piece of metadata helps with search engine optimisation
`viewport`	Information about the viewport, i.e. the window in which the user sees the webpage. For example, it is common to include the following element in webpages to improve their viewability across different devices: `<meta name="viewport" content="width=device-width, initial-scale=1.0">` The `width=device-width` part of the content attribute indicates that the width of the page should adapt to the screen width, and the `initial-scale=1.0` part identifies the initial zoom level used when the page is first loaded into the browser.

novalidate

[HTMLAttributeNovalidate]

The HTML `novalidate` attribute indicates whether a <form> element should not be validated when submitted.

Valid attribute values (for <form>) include:

Value	Description
`novalidate`	Whether form-data (i.e. input) should not be validated when submitted

onabort

[HTMLAttributeOnabort]

The HTML `onabort` attribute specifies the event that is triggered if the document is aborted. It applies to <audio>, <embed>, <img>, <object> and <video> elements.

onafterprint

[HTMLAttributeOnafterprint]

The HTML `onafterprint` attribute specifies the event that is triggered after a document is printed. It applies to <body> elements.

onbeforeprint

[HTMLAttributeOnbeforeprint]

The HTML `onbeforeprint` attribute specifies the event that is triggered before a document is printed. It applies to <body> elements.

onbeforeunload

[HTMLAttributeOnbeforeunload]

The HTML `onbeforeunload` attribute specifies the event that is triggered just before a document is unloaded. It applies to <body> elements.

It can be used to return a message if the user is just about to leave the page.

onblur

[HTMLAttributeOnblur]

The HTML `onblur` attribute specifies the event that is triggered when an element loses focus. It applies to all visible elements.

oncanplay

[HTMLAttributeOncanplay]

The HTML `oncanplay` attribute specifies the event that is triggered when a file is ready to start playing (i.e. when it has buffered enough to begin). It applies to <audio>, <embed>, <object> and <video> elements.

oncanplaythrough

[HTMLAttributeOncanplaythrough]

The HTML `oncanplaythrough` attribute specifies the event that is triggered when a file is ready to play all the way to its end without pausing for buffering. It applies to <audio> and <video> elements.

onchange

[HTMLAttributeOnchange]

The HTML `onchange` attribute specifies the event that is triggered when an element's value changes. It applies to all visible elements.

onclick

[HTMLAttributeOnclick]

The HTML `onclick` attribute specifies the event that is triggered when an element is clicked (mouse clicked). It applies to all visible elements.

oncontextmenu

[HTMLAttributeOncontextmenu]

The HTML `oncontextmenu` attribute specifies the event that is triggered when a context menu is triggered. It applies to all visible elements.

oncopy

[HTMLAttributeOncopy]

The HTML `oncopy` attribute specifies the event that is triggered when the content of an element is copied. It applies to all visible elements.

oncuechange

[HTMLAttributeOncuechange]

The HTML `oncuechange` attribute specifies the event that is triggered when the cue changes in a <track> element. It applies to <track> elements.

oncut

[HTMLAttributeOncut]

The HTML `oncut` attribute specifies the event that is triggered when the content of an element is cut. It applies to all visible elements.

ondblclick

[HTMLAttributeOndblclick]

The HTML `ondblclick` attribute specifies the event that is triggered when an element is double-clicked (mouse double-clicked). It applies to all visible elements.

ondrag

[HTMLAttributeOndrag]

The HTML `ondrag` attribute specifies the event that is triggered when an element is dragged. It applies to all visible elements.

ondragend

[HTMLAttributeOndragend]

The HTML `ondragend` attribute specifies the event that is triggered at the end of a drag operation. It applies to all visible elements.

ondragenter

[HTMLAttributeOndragenter]

The HTML `ondragenter` attribute specifies the event that is triggered when an element has been dragged to a valid drop target. It applies to all visible elements.

ondragleave

[HTMLAttributeOndragleave]

The HTML `ondragleave` attribute specifies the event that is triggered when an element is dragged outside a valid drop target. It applies to all visible elements.

ondragover

[HTMLAttributeOndragover]

The HTML `ondragover` attribute specifies the event that is triggered when an element is being dragged over a valid drop target. It applies to all visible elements.

ondragstart

[HTMLAttributeOndragstart]

The HTML `ondragstart` attribute specifies the event that is triggered at the start of a drag. It applies to all visible elements.

ondrop

[HTMLAttributeOndrop]

The HTML `ondrop` attribute specifies the event that is triggered when a dragged element is dropped. It applies to all visible elements.

ondurationchange

[HTMLAttributeOndurationchange]

The HTML `ondurationchange` attribute specifies the event that is triggered when the length of a media changes. It applies to <audio> and <video> elements.

onemptied

[HTMLAttributeOnemptied]

The HTML `onemptied` attribute specifies the event that is triggered when a media file unexpected becomes unavailable. It applies to <audio> and <video> elements.

onended

[HTMLAttributeOnended]

The HTML `onended` attribute specifies the event that is triggered when a media reaches its end. It applies to <audio> and <video> elements.

onerror

[HTMLAttributeOnerror]

The HTML `onerror` attribute specifies the event that is triggered when an error occurs. It applies to <audio>, <body>, <embed>, <img>, <object>, <script>, <style> and <video> elements.

onfocus

[HTMLAttributeOnfocus]

The HTML `onfocus` attribute specifies the event that is triggered when an element gets focus. It applies to all visible elements.

onfocusin

[HTMLAttributeOnfocusin]

The HTML `onfocusin` attribute specifies the event that is triggered when an element is about to get focus. It is similar to the onfocus attribute except that it also 'bubbles'.

onfocusout

[HTMLAttributeOnfocusout]

The HTML `onfocusout` attribute specifies the event that is triggered when an element is about to lose focus. It is similar to the onblur attribute except that it also 'bubbles'.

onhashchange

[HTMLAttributeOnhashchange]

The HTML `onhashchange` attribute specifies the event that is triggered when there is a change to the anchor part of a URL (i.e. the part after a #). It applies to <body> elements.

oninput

[HTMLAttributeOninput]

The HTML `oninput` attribute specifies the event that is triggered when an element gets user input. It applies to all visible elements.

oninvalid

[HTMLAttributeOninvalid]

The HTML `oninvalid` attribute specifies the event that is triggered when an element is invalid. It applies to all visible elements.

onkeydown

[HTMLAttributeOnkeydown]

The HTML `onkeydown` attribute specifies the event that is triggered when the user is pressing a key. It applies to all visible elements.

onkeypress

[HTMLAttributeOnkeypress]

The HTML `onkeypress` attribute specifies the event that is triggered when the user presses a key. It applies to all visible elements.

onkeyup

[HTMLAttributeOnkeyup]

The HTML `onkeyup` attribute specifies the event that is triggered when the user releases a key. It applies to all visible elements.

onload

[HTMLAttributeOnload]

The HTML `onload` attribute specifies the event that is triggered when an element finishes loading. It applies to <body>, <iframe>, <img>, <input>, <link>, <script> and <style> elements.

onloadeddata

[HTMLAttributeOnloadeddata]

The HTML `onloadeddata` attribute specifies the event that is triggered when data for the current frame is loaded, but not enough data is yet loaded to play the next frame. It applies to <audio> and <video> elements.

onloadedmetadata

[HTMLAttributeOnloadedmetadata]

The HTML `onloadedmetadata` attribute specifies the event that is triggered when metadata (dimensions, duration, …) is loaded. It applies to <audio> and <video> elements.

onloadstart

[HTMLAttributeOnloadstart]

The HTML `onloadstart` attribute specifies the event that is triggered just before loading starts. It applies to <audio> and <video> elements.

During loading the following events occur in the following order:

- onloadstart
- ondurationchange
- onloadedmetadata
- onloadeddata
- onprogress
- oncanplay
- oncanplaythrough

onmessage

[HTMLAttributeOnmessage]

The HTML `onmessage` attribute specifies the event that is triggered when a message is received through an event source.

onmousedown

[HTMLAttributeOnmousedown]

The HTML `onmousedown` attribute specifies the event that is triggered when the mouse button is pressed down on an element. It applies to all visible elements.

onmouseenter

[HTMLAttributeOnmouseenter]

The HTML `onmouseenter` attribute specifies the event that is triggered when the mouse pointer moves onto an element. It applies to all visible elements. It is often used in conjunction with the onmouseleave event.

It is like the onmouseover event (or the onmousemove event), except that the `onmouseenter` event only fires when the mouse first enters the element itself, whereas the onmouseover event also fires in response to the mouse moving into the element from a child element that is located within the original element.

onmouseleave

[HTMLAttributeOnmouseleave]

The HTML `onmouseleave` attribute specifies the event that is triggered when the mouse pointer moves onto an element. It applies to all visible elements. It is often used in conjunction with the onmouseenter event.

It is like the onmouseout event (or the onmousemove event), except that the `onmouseleave` event only fires when the mouse first leaves the element itself, whereas the onmouseout event also fires in response to the mouse moving out of the element into a child element that is located within the original element.

onmousemove

[HTMLAttributeOnmousemove]

The HTML `onmousemove` attribute specifies the event that is triggered for as long as the mouse pointer is moving over an element. It applies to all visible elements.

onmouseout

[HTMLAttributeOnmouseout]

The HTML `onmouseout` attribute specifies the event that is triggered when the mouse pointer moves outside an element. It applies to all visible elements.

onmouseover

[HTMLAttributeOnmouseover]

The HTML `onmouseover` attribute specifies the event that is triggered when the mouse pointer moves over an element. It applies to all visible elements.

onmouseup

[HTMLAttributeOnmouseup]

The HTML `onmouseup` attribute specifies the event that is triggered when the mouse pointer is released over an element. It applies to all visible elements.

onmousewheel

[HTMLAttributeOnmousewheel]

The HTML `onmousewheel` attribute specifies the event that is triggered when the mouse pointer is released over an element. It applies to all visible elements. Depreciated, use onwheel instead.

onoffline

[HTMLAttributeOnoffline]

The HTML `onoffline` attribute specifies the event that is triggered when the browser starts to work offline. It applies to <body> elements.

ononline

[HTMLAttributeOnonline]

The HTML `ononline` attribute specifies the event that is triggered when the browser starts to work online. It applies to <body> elements.

onopen

[HTMLAttributeOnopen]

The HTML `onopen` attribute specifies the event that is triggered when a connection to an event source is opened.

onpagehide

[HTMLAttributeOnpagehide]

The HTML `onpagehide` attribute specifies the event that is triggered when the user navigates away from a page. It applies to <body> elements.

onpageshow

[HTMLAttributeOnpageshow]

The HTML `onpageshow` attribute specifies the event that is triggered when the user navigates to a page. It applies to <body> elements.

onpaste

[HTMLAttributeOnpaste]

The HTML `onpaste` attribute specifies the event that is triggered when the user pastes content in an element. It applies to all visible elements.

onpause

[HTMLAttributeOnpause]

The HTML `onpause` attribute specifies the event that is triggered when a media is paused. It applies to <audio> and <video> elements.

onplay

[HTMLAttributeOnplay]

The HTML `onplay` attribute specifies the event that is triggered when a media is ready to start playing. It applies to <audio> and <video> elements.

onplaying

[HTMLAttributeOnplaying]

The HTML `onplaying` attribute specifies the event that is triggered when a media has started playing. It applies to <audio> and <video> elements.

onpopstate

[HTMLAttributeOnpopstate]

The HTML `onpopstate` attribute specifies the event that is triggered when the window's history changes. It applies to <body> elements.

onprogress

[HTMLAttributeOnprogress]

The HTML `onprogress` attribute specifies the event that is triggered when the browser is in the process of getting media data. It applies to <audio> and <video> elements.

onratechange

[HTMLAttributeOnratechange]

The HTML `onratechange` attribute specifies the event that is triggered when the playback rate of a media changes (e.g. the user switches to fast forward). It applies to <audio> and <video> elements.

onreset

[HTMLAttributeOnreset]

The HTML `onreset` attribute specifies the event that is triggered when the reset button in a form is clicked. It applies to <form> elements.

onresize

[HTMLAttributeOnresize]

The HTML `onresize` attribute specifies the event that is triggered when the browser window is being resized. It applies to <body> elements.

onscroll

[HTMLAttributeOnscroll]

The HTML `onscroll` attribute specifies the event that is triggered when the element's scrollbar is being scrolled. It applies to all visible elements.

onsearch

[HTMLAttributeOnsearch]

The HTML `onsearch` attribute specifies the event that is triggered when user enters something in a search field (for an <input> element of type = `search`). It applies to <input> elements.

onseeked

[HTMLAttributeOnseeked]

The HTML `onseeked` attribute specifies the event that is triggered when the seeking attribute of a media is set to false (i.e. the seeking has finished). It applies to <audio> and <video> elements.

onseeking

[HTMLAttributeOnseeking]

The HTML `onseeking` attribute specifies the event that is triggered when the seeking attribute of a media is set to true (i.e. the seeking is active). It applies to <audio> and <video> elements.

onselect

[HTMLAttributeOnselect]

The HTML `onselect` attribute specifies the event that is triggered when an element gets selected. It applies to all visible elements.

onshow

[HTMLAttributeOnshow]

The HTML `onshow` attribute specifies the event that is triggered when a <menu> element is shown as a context menu. It applies to <menu> elements.

onstalled

[HTMLAttributeOnstalled]

The HTML `onstalled` attribute specifies the event that is triggered when the browser is unable to fetch the media data (for whatever reason). It applies to <audio> and <video> elements.

onstorage

[HTMLAttributeOnstorage]

The HTML `onstorage` attribute specifies the event that is triggered when the web storage area is updated. It applies to <body> elements.

onsubmit

[HTMLAttributeOnsubmit]

The HTML `onsubmit` attribute specifies the event that is triggered when a form is submitted. It applies to <form> elements.

onsuspend

[HTMLAttributeOnsuspend]

The HTML `onsuspend` attribute specifies the event that is triggered when the fetching of media data is stopped before completely loaded (for whatever reason). It applies to <audio> and <video> elements.

ontimeupdate

[HTMLAttributeOntimeupdate]

The HTML `ontimeupdate` attribute specifies the event that is triggered when the playing position in a media has changed (e.g. user fast forwards to a new position in the media). It applies to <audio> and <video> elements.

ontoggle

[HTMLAttributeOntoggle]

The HTML `ontoggle` attribute specifies the event that is triggered when the user opens or closes a <details> element. It applies to <details> elements.

ontouchcancel

[HTMLAttributeOntouchcancel]

The HTML `ontouchcancel` attribute specifies the event that is triggered when touch is interrupted. It applies to touch-sensitive elements.

ontouchend

[HTMLAttributeOntouchend]

The HTML `ontouchend` attribute specifies the event that is triggered when the touching device (usually a finger) is removed from the touch screen. It applies to touch-sensitive elements.

ontouchmove

[HTMLAttributeOntouchmove]

The HTML `ontouchmove` attribute specifies the event that is triggered when the touching device (usually a finger) is dragged across the touch screen. It applies to touch-sensitive elements.

ontouchstart

[HTMLAttributeOntouchstart]

The HTML `ontouchstart` attribute specifies the event that is triggered when the touching device (usually a finger) is placed on the touch screen. It applies to touch-sensitive elements.

onunload

[HTMLAttributeOnunload]

The HTML `onunload` attribute specifies the event that is triggered when the page has unloaded (or the browser window has closed). It applies to <body> elements.

onvolumechange

[HTMLAttributeOnvolumechange]

The HTML `onvolumechange` attribute specifies the event that is triggered when the volume of a media is changed (or muted). It applies to <audio> and <video> elements.

onwaiting

[HTMLAttributeOnwaiting]

The HTML `onwaiting` attribute specifies the event that is triggered when the media has paused but is expected to resume (e.g. the media has paused to buffer more data). It applies to <audio> and <video> elements.

onwheel

[HTMLAttributeOnwheel]

The HTML `onwheel` attribute specifies the event that is triggered when the mouse wheel rolls up or down over an element. It applies to all visible elements. It currently is not supported by all major browsers.

open

[HTMLAttributeOpen]

The HTML `open` attribute indicates whether details in a <details> or <dialog> element should be visible (i.e. open) to the user.

Valid attribute values (for <details>) include:

Value	Description
`open`	Specifies whether the details should be visible to user

Valid attribute values (for <dialog>) include:

Value	Description
`open`	Specifies whether the dialog element is active and hence whether the user can interact with it

optimum

[HTMLAttributeOptimum]

The HTML `optimum` attribute indicates the value that is deemed optimal for the gauge applicable to a <meter> element.

Valid attribute values (when used with <meter> elements) include:

Value	Description
number	(Floating point) number defining optimal value for gauge

pattern

[HTMLAttributePattern]

The HTML `pattern` attribute indicates the format expression that the value of an <input> element is checked against.

Valid attribute values (when used with <input> elements) include:

Value	Description
regular-expression	Regular expression against which input is compared

placeholder

[HTMLAttributePlaceholder]

The HTML `placeholder` attribute indicates the short hint describing the expected value of an <input> or <textarea> element.

Valid attribute values (when used with <input> and <textarea> elements) include:

Value	Description
text	A short hint that describes what is expected as input

poster

[HTMLAttributePoster]

The HTML `poster` attribute indicates the image to be shown while a <video> element is downloading (or until user hits play).

Valid attribute values (when used with <video> elements) include:

Value	Description

URL	URL of file containing image to be shown whilst video is downloading or until user hits play button

preload

[HTMLAttributePreload]

The HTML `preload` attribute indicates if / how the page author thinks <audio> or <video> elements should be loaded when the page loads.

Valid attribute values (when used with <audio> and <video> elements) include:

Value	Description
`auto`	Browser to load entire video when page loads
`metadata`	Browser should only load the video's metadata
`none`	Browser should not load video when page loads

radiogroup

[HTMLAttributeRadiogroup]

The HTML `radiogroup` attribute specifies the commands that are toggled when a <menuitem> element is toggled.

Valid attribute values (when used with <menu> elements) include:

Value	Description
groupname	Name of group of commands toggled when the menu item is toggled. Only applies if type = `radio`

readonly

[HTMLAttributeReadonly]

The HTML `readonly` attribute indicates whether an <input> or <textarea> element is read-only.

Valid attribute values (when used with <input> and <textarea> elements) include:

Value	Description
`readonly`	Indicates that input box is read-only

rel

[HTMLAttributeRel]

The HTML `rel` attribute indicates the relationship between the current document and the document to which it is linked. It applies to <a>, <area> and <link> elements.

Valid attribute values (when used with <a>, <area> and <link> elements) include:

Value	Description
alternate	An alternative representation of the document
author	Link to resource describing author of document
bookmark	URL used for bookmarking (for <a>and <area>)
dns-prefetch	Browser should preemptively do a DNS on the origin of the target (for <link>)
external	Referenced document is not part of same site as original (for <a>)
help	Document providing help
icon	Get icon representing document (for <link>)
license	Copyright information on the document
next	Next document in series
nofollow	An unendorsed document (e.g. a paid link, search spiders may not then follow that link)
noreferrer	Browser should not send an HTTP referrer header if user follows hyperlink
noopener	Browser context created by following hyperlink should not have an opener browser context (for <a>)
pingback	Address of pingback server handling pingbacks relating to origin of document (for <link>)
preconnect	Browser should pre-emptively connect to target (for <link>)
prefetch	Browser should pre-emptively fetch and cache target (for <area>, <link>)
preload	Browser should pre-emptively fetch and render target (for <link>)
prerender	Same as preload for some browsers (for <link>)
prev	Previous document in series
search	A search tool covering the document
stylesheet	Import a CSS stylesheet (for <link>)
tag	A keyword relevant to the current document (for <a>and <area>)

required

[HTMLAttributeRequired]

The HTML required attribute indicates whether an <input>, <select> or <textarea> element needs to be filled in before a form is submitted.

Valid attribute values (when used with <input>, <select> and <textarea> elements) include:

Value	Description
required	Indicates that input box or selection choice must be filled out before the form can be submitted

reversed

[HTMLAttributeReversed]

The HTML reversed attribute indicates whether the list order of an <ol> element should be in descending order (e.g. 3, 2, 1) rather than ascending order (1, 2, 3).

Valid attribute values (when used with <ol> elements) include:

Value	Description

`reversed`	List order is descending

rows

[HTMLAttributeRows]

The HTML `rows` attribute indicates the visible number of lines in a <textarea> element.

Valid attribute values (when used with <textarea> elements) include:

Value	Description
integer	Visible number of rows (lines) in text area

rowspan

[HTMLAttributeRowspan]

The HTML `rowspan` attribute indicates the number of rows a table cell should span. It applies to <td> and <th> elements.

Valid attribute values (when used with <td> and <th> elements) include:

Value	Description
integer	Number of rows a cell should span

sandbox

[HTMLAttributeSandbox]

The HTML `sandbox` attribute indicates extra restrictions applied to the content of an <iframe> element. The sorts of additional restrictions that can be imposed include:

- Deeming the content to come from a unique origin
- Blocking form submission, script execution or execution of APIs
- Preventing some sorts of links
- Preventing content from using plug-ins (e.g. from <embed> or <object> elements)
- Blocking some automatically triggered features (such as automatically playing a <video> element)

Valid attribute values (when used with <iframe> elements) include either `sandbox` (which results in all restrictions being applied) or a space delimited list of values that exclude specific restrictions. These values are:

Value	Description
`allow-forms`	Form submission enabled
`allow-pointer-lock`	APIs allowed
`allow-popups`	Popups allowed
`allow-same-origin`	<iframe> content allowed to be treated as being from same origin as main document

`allow-scripts`	Scripts allowed
`allow-top-navigation`	<iframe> content allowed to navigate to its top-level browsing context
`sandbox` (i.e. *no value*)	All restrictions applied

scope

[HTMLAttributeScope]

The HTML `scope` attribute indicates whether a table header cell (i.e. a <th> element) is a header for a column, a row or for groups of either columns or rows. The `scope` attribute is no longer supported in HTML 5.

Valid attribute values (when used with <td> elements) include:

Value	Description
`col`	Cell is header for a column
`colgroup`	Cell is header for a group of columns
`row`	Cell is header for a row
`rowgroup`	Cell is header for a group of rows

scoped

[HTMLAttributeScoped]

The HTML `scoped` attribute indicates that styles in a <style> element only apply to the element's parent and that element's child elements. The aim is to allow style declarations that apply only to individual parts of a HTML document, but currently it does not always work with major browsers.

Valid attribute values (when used with <style> elements) include:

Value	Description
`scoped`	Styles only apply to element's parent element and that element's child elements

selected

[HTMLAttributeSelected]

The HTML `selected` attribute indicates that an <option> element should be pre-selected when the page loads.

Valid attribute values (when used with <option> elements) include:

Value	Description
`selected`	Option should be pre-selected when page loads

shape

[HTMLAttributeShape]

The HTML `shape` attribute indicates shape of an <area> element. It, together with the coords attribute specify the size, shape and position of the area.

Valid attribute values (applied to <area> elements) include:

Value	Description
`default`	Indicates the entire region
`circle`	Indicates a circular region
`poly`	Indicates a polygonal region
`rect`	Indicates a rectangular region

size

[HTMLAttributeSize]

The HTML `size` attribute indicates the width in characters for <input> elements or number of visible options for <select> elements.

Valid attribute values (when used with <input> elements) include:

Value	Description
integer	Number of characters that identify the width of the element

Valid attribute values (when used with <select> elements) include:

Value	Description
integer	Number of visible options in drop-down menu

sizes

[HTMLAttributeSizes]

The HTML `sizes` attribute specifies the size of a linked resource. It applies to <img>, <link> and <source> elements.

Valid attribute values (when used with <link> elements) include:

Value	Description
heightxwidth	Specifies one or more sizes for linked icon, in the form e.g. `sizes="16x16"` or `sizes="16x16 32x32"`. Is only relevant for `rel=icon`
`any`	Icon is scalable

Note: most browsers do not currently seem to support the `sizes` attribute (at the time of writing it was an experimental attribute for <link> elements). For <source> elements it is only relevant when the <source> element is a direct child of a picture element.

span

[HTMLAttributeSpan]

The HTML `span` attribute specifies the number of columns that a <col> or <colgroup> element spans.

Valid attribute values (for <col>, <colgroup>) include:

Value	Description
integer	Number of columns the element should span

spellcheck

[HTMLAttributeSpellcheck]

The HTML `spellcheck` attribute indicates whether an element is to have its spelling and grammar checked.

Valid attribute values include:

Value	Description
`false`	Element is not to be spellchecked
`true`	Element is to be spellchecked and grammar-checked

src

[HTMLAttributeSrc]

The HTML `src` attribute indicates the URL of a resource. It applies to <audio>, <embed>, <iframe>, <img>, <input>, <script>, <source>, <track> and <video> elements.

Valid attribute values (when used with <audio>, <embed>, <iframe>, <img>, <script>, <source>, <track> and <video> elements) include:

Value	Description
URL	URL of source

Valid attribute values (when used with <input> elements) include:

Value	Description
URL	URL of image to use as a submit button (only for type = `image`)

srcdoc

[HTMLAttributeSrcdoc]

The HTML `srcdoc` attribute indicates the HTML content of the page to be shown in an <iframe> element.

If a browser supports this attribute then it will override the content specified by the src attribute (if present). If it does not support this attribute then it will show the file specified by the src attribute (if present).

The `srcdoc` attribute is usually used in conjunction with the sandbox attribute and the seamless attribute (the seamless attribute is not currently supported by major browsers so is not covered further here).

Valid attribute values (when used with <iframe> elements) include:

Value	Description
HTML_content	HTML content of the page to show in the element

srclang

[HTMLAttributeSrclang]

The HTML `srclang` attribute indicates the language of text data of a <track> element if its `kind` = `subtitles`.

Valid attribute values (when used with <input> elements) include:

Value	Description
language-code	Language of track text data (only for kind = `subtitles`)

srcset

[HTMLAttributeSrcset]

The HTML `srcset` attribute indicates the URL of image to use in different situations for <img> and <source> elements.

Valid attribute values (when used with <img> and <source> elements) include:

Value	Description
URL	URL of image to use in different situations

start

[HTMLAttributeStart]

The HTML `start` attribute indicates the Start value to use for an ordered list (i.e. an <ol> element).

Valid attribute values (when used with <ol> elements) include:

Value	Description
integer	Starting value of list

step

[HTMLAttributeStep]

The HTML `step` attribute indicates the accepted number intervals for an <input> element. For example, if `step="4"` then the accepted numbers could be -4, 0, 4, 8, ….

Valid attribute values (when used with <input> elements) include:

Value	Description
integer	Number of intervals

style

[HTMLAttributeStyle]

The HTML `style` attribute indicates an 'inline' CSS style for that element.

Valid values include:

Value	Description
CSSstyle	A CSS style definition

tabindex

[HTMLAttributeTabindex]

The HTML `tabindex` attribute indicates the tab order of an element, i.e. the order in which the user is taken between elements when pressing the tab key (1 is first element).

In HTML 5 this attribute can be applied to any element.

Valid attribute values include:

Value	Description
integer	Tab order

target

[HTMLAttributeTarget]

The HTML `target` attribute indicates where / how to open a linked document (or where to submit a form). It applies to <a>, <area>, <base> and <form> elements.

Valid attribute values (for <a>, <area>, <base> and <form>) include:

Value	Description
`_blank`	Opens linked document in a new window or tab
`_parent`	Opens linked document in parent frame
`_self`	(default value). Opens linked document in the same window or tab as was clicked
`_top`	Opens linked document in full body of the window

framename	Opens linked document in named frame (not applicable to <form> elements)

title

[HTMLAttributeTitle]

The HTML `title` attribute specifies extra information about an element. Often, if you move the mouse over an element with its title set then the title typically appears as tooltip text next to the element.

In HTML 5 this attribute can be applied to any element.

Valid attribute values include:

Value	Description
text	Tooltip text relating to element

transitionend

[HTMLAttributeTransitionend]

The HTML `transitionend` attribute specifies the event that is triggered when a CSS transition ends. It applies to any element with a CSS transition.

translate

[HTMLAttributeTranslate]

The HTML `translate` attribute specifies whether the content of an element should be translated.

Valid attribute values include:

Value	Description
`yes`	Element content should be translated
`no`	Element content should not be translated

Note: at the time of writing this attribute was not supported by major browsers.

type

[HTMLAttributeType]

The HTML `type` attribute indicates the type of an element. It applies to <area>, <button>, <embed>, <input>, <keygen>, <link>, <menu>, <menuitem>, <object>, <ol>, <script>, <source> and <style> elements.

Valid attribute values (when used with <area>, <embed>, <link>, <object>, <script> elements) include:

Value	Description
media_type	The internet media type (previously known as MIME type) of the target URL

Valid attribute values (when used with <button> elements) include:

Value	Description
`button`	Is a clickable button
`reset`	Is a submit button (so submits form data)
`submit`	Is a reset button (so resets form data to initial / default values)

Valid attribute values (when used with <input> and <keygen> elements) include:

Value	Description
`button`	A clickable button that (typically) activates some JavaScript when clicked
`checkbox`	Input field allowing selection of one or more options from a limited list of options
`color`	Input field for selecting a colour
`date`	Input field for entering a date
`datetime`	Input field for entering a date and time (including time zone) (N.B. Is not currently supported by most browsers)
`datetime-local`	Input field for entering a date and time (no specific time zone)
`email`	E-mail address input field (automatically validated when submitted)
`file`	Define a file selection (with browse) button (for file uploads)
`hidden`	Hidden field (i.e. not visible to user). Is often used to store a default value or may be used as a variable by JavaScript
`image`	Define image as a submit button
`month`	Input field for entering a month and year (no specific time zone)
`number`	Input field for entering a number. The default value is the value attribute. Minimum, maximum and legal number intervals are defined by the min, max and step attributes
`password`	Password field (characters are masked)
`radio`	Buttons allowing user to select only one of a limited number of choices
`range`	Slider control, i.e. a control whose exact values are not important. Default range is 0 to 100 but restrictions can be placed, e.g. using the min, max and step attributes.
`reset`	Reset button (e.g. for resetting all form values to default values)
`search`	A search field
`submit`	Submit button
`tel`	Input field for entering a telephone number
`text`	Single-line input field for entering text
`time`	Input field for entering a time (no specific time zone)
`url`	Input field for entering a URL
`week`	Input field for entering a week and year (no specific time zone)

Valid attribute values (when used with <menu> elements) include:

Value	Description

`list`	A list menu
`toolbar`	A toolbar menu
`context`	A context menu

Valid attribute values (when used with <menuitem> elements) include:

Value	Description
`checkbox`	Command can be toggled using a checkbox
`command`	A normal command
`Radio`	Command can be toggled using a radio button

Note: at the time of writing the `type` attribute for <menuitem> elements was not supported by major browsers.

Valid attribute values (when used with <ol> elements) include:

Value	Description
`1`	List is of type 1, 2, 3, 4, ...
`A`	List is of type A, B, C, D, ...
`a`	List is of type a, b, c, d, ...
`I`	List is of type I, II, III, IV, ...
`i`	List is of type i, ii, iii, iv, ...

Valid values (when used with <source> elements) include:

Value	Description
MIME-type	MIME type of resource

Valid attribute values (when used with <style> elements) include:

Value	Description
`text/css`	Media type of the <style> element

usemap

[HTMLAttributeUsemap]

The HTML `usemap` attribute indicates whether an <img> or <object> element should be used as a client-side image-map.

Valid attribute values (when used with <img> or <object> elements) include:

Value	Description
#mapname	Name of client-side image-map, i.e. a hash character (#) plus the name of the <map> element to which the `usemap` relates

value

[HTMLAttributeValue]

The HTML `value` attribute indicates the value of an element. It applies to <button>, <data>, <input>, <li>, <meter>, <option>, <progress> and <param> elements.

Valid attribute values (when used with <button>, <input> elements) include:

Value	Description
text	Initial value for button

Note: for some older browsers, if you use a <button> element inside a <form> element then the browser may submit the text between the `<button>` and `</button>` tags rather than the value of its `value` attribute.

Valid attribute values (when used with <data> elements) include:

Value	Description
machine-readable format	Machine-readable content

Valid attribute values (when used with <li> elements) include:

Value	Description
integer	Value of a list item (following list items will increment from that number). Only applicable to lists

Valid attribute values (when used with <meter> elements) include:

Value	Description
number	Value of gauge

Valid attribute values (when used with <option> elements) include:

Value	Description
text	Value to be sent to server

width

[HTMLAttributeWidth]

The HTML `width` attribute indicates the width of an element. It applies to <canvas>, <embed>, <iframe>, <img>, <input>, <object> and <video> elements.

Valid attribute values (when used with <canvas>, <embed>, <iframe>, <img>, <input>, <object> and <video> elements) include:

Value	Description
number	Width of element or embedded content in pixels, e.g. `width="20"`
percentage	Width as a percentage of surrounding element, e.g. `width="30%"`

Note: for some browsers and for some (but not all) of the elements listed above it appears to be necessary if the width is being set in JavaScript to set it using the CSS width property, e.g. in the form *element*`.style.width = "20px"` rather than using the `width` attribute.

wrap

[HTMLAttributeWrap]

The HTML `wrap` attribute indicates how text in a <textarea> element is to be wrapped when submitted in a form.

Valid attribute values (when used with <textarea> elements) include:

Value	Description
`hard`	Text is wrapped (contains newlines) when submitted. The cols attribute must then be specified
`soft`	(default value). Text is not wrapped when submitted

xmlns

[HTMLAttributeXmlns]

The HTML `xmlns` attribute indicates the XML namespace attribute applicable to the webpage (if it needs to conform to XHTML). It applies to <html> elements.

Valid attribute values (when used with <html> elements) include:

Value	Description
`http://www.w3.org/1999/xhtml`	The default XHTML specification

HTML: types of attribute values

[HTMLTypesOfAttributeValues]

Many HTML attributes accept specific types of input, including the following:

Value	Description
#mapname	Name of client-side image-map, i.e. a hash character (#) plus the name of a <map> element
character	A single keyboard character (e.g. as per the accesskey attribute)
character_set	A character encoding (i.e. way of representing characters that will be recognised by a receiving computer), such as: - `UTF-8`: Unicode - `ISO-8859-1`: character encoding for Latin alphabet
CSSclass	Name of a CSS class. These must begin with a letter A-Z or a-z, which can be followed by letters (A-Z or a-z), digits (0-9), hypens ("-") and underscores ("_"). In HTML all such values are case-insensitive, i.e. class names of "abc" and "ABC" are treated as synonymous.
CSSstyle	A CSS style definition
datalist_id	Id (identifier) of relevant <datalist> element
date	A date
elementID	The id (identifier) defining the associated element
file_extension	A file extension starting with a full stop, e.g. `.png`, `.jpg`, `.pdf`, `.doc`

	(e.g. used for the accept attribute)
filename	File name of a resource
formID	The id defining the associated form
framename	A named frame (i.e. <iframe> element)
groupname	Name of group of commands
header_id	Id of a header cell
heightxwidth	One or more sizes (in pixels), in the form e.g. `sizes="16x16"` or `sizes="16x16 32x32"`.
HTML_content	HTML content
HTTPmethod	Either get or post. These have the following characteristics: - `get`. Use the HTTP 'get' method. This includes the form-data in the URL in name/value pairs. The length of the URL is limited and hence the values transmitted will be public (even if the website is accessed using an https call), but users can bookmark the resulting call - `post`. Use the HTTP 'post' method. This includes the form-data in the HTTP request, i.e. not in the URL, and is not subject to the same size limitations as the 'get' method, but cannot then be bookmarked by users
inputfieldname	Name of an input field
integer	An integer
language-code	Language of text in linked document. The language code is either an ISO 639-1 two letter language code (e.g. "en") or such a code followed by a dash and then a two letter ISO country code (the latter can be used if different countries recognise different versions of the same language, e.g. "en-gb" versus "en-us")
machine-readable format	Machine-readable content
media-query	Media or device type
media_type	A valid media type, see e.g. http://www.iana.org/assignments/media-types/media-types.xhtml (e.g. as per accept attribute)
MIME-type	MIME type of resource
name	Name of element, attribute or (for <meta> elements) metadata item
no value	I.e. element should be left black, however see below regarding attribute minimisation and XHTML.
number	A number (sometimes only an integer is acceptable, e.g. for the cols attribute, sometimes a floating-point value is also acceptable), usually in the form of a string (enclosed in quotes) representing the number
percentage	A percentage, e.g. `30%`
regular-expression	Regular expression (against which e.g. an input is compared)
text	Text
URI	Uniform Resource Identifier, see below.
URL	i.e. Uniform Resource Locator. These can be: - Absolute, pointing to a specific webpage, e.g. http://www.cnsglobal.tech, or - Relative, pointing to a file relative to some base, usually the directory or website within which the page accessing the URL is position, e.g. example.htm
x1, *y1*, *x2*, *y2*	Typically involve coordinates as per the coords attribute
YYYY-MM-	A date (in a specific machine and location independent format)

DDThh:mm:ssTZD	

When a value the attribute can take is shown as the same as its own name then this is a Boolean-style attribute, meaning that in HTML the attribute should either be mentioned (but assigned no value), in which case the attribute applies, or be absent, in which case the attribute does not apply. In XHTML, such *attribute minimisation* is not allowed, and the attribute needs to be defined explicitly, taking as its value its own name, e.g. `<video … autoplay="autoplay">…</video>`.

Event attributes (which usually have the form `on…`) take values which are JavaScript functions.

Uniform Resource Identifiers (URIs):

'URI' stands for 'Uniform Resource Identifier'. The possible set of parts a URI can contain are illustrated by the following:

```
http://username:pword@www.example.org:80/path/file.aspx?a=23&b=has+spaces#anchor
```

A URI encoded string has each instance of certain characters replaced by one, two, three or (rarely) four escape sequences representing the UTF-8 encoding of the character. `encodeURI` escapes all characters except for the following (so it does not encode characters needed to formulate a complete URI as above, or a few additional 'unreserved marks' which do not have a reserved purpose as such and are allowed in a URI 'as is'):

```
A-Z a-z 0-9 ; , / ? : @ & = + $ - _ . ! ~ * ' ( ) #
```

There are four global JavaScript methods that convert strings into URIs and vice versa (six if depreciated methods are included).

`encodeURI()` escapes all characters except for the following (so it does not encode characters needed to formulate a complete URI as above, or a few additional 'unreserved marks' which do not have a reserved purpose as such and are allows in a URI 'as is':

```
A-Z a-z 0-9 ; , / ? : @ & = + $ - _ . ! ~ * ' ( ) #
```

`encodeURIComponent()` also escapes reserved characters, so escapes all characters except:

```
A-Z a-z 0-9 - _ . ! ~ * ' ( ) #
```

`escape()` (now depreciated, use `encodeURI` or `encodeURIComponent` instead) encodes all characters with the exception of `* @ - _ + . /`

`decodeURI()`, `decodeURIComponent()` and `unescape()` are the inverses of `encodeURI()`, `encodeURIComponent()` and `escape()` respectively.

If you want to encode a string but avoid encoding square brackets (these are becoming reserved characters for IPv6) then it is recommended that you use a JavaScript statement like:

```
encode(str).replace(/%5B/g,'[').replace(%5D/g,']')
```

Appendix C: CSS Properties

[CSSProperties]

Set out below are different CSS properties (and rules):

Property	Description	More	Type
align-content	Modifies the behaviour of the flex-wrap property, aligning flex lines	Here	Flexible Container
align-items	Specifies the default alignment for items inside a flexible container	Here	Flexible Container
align-self	Specifies the alignment for selected item in a flexible container	Here	Flexible Container
all	Resets (almost) all properties to their initial or inherited values	Here	All
animation	A shorthand property combining (up to) 8 individual animation properties	Here	Animation
animation-delay	Specifies delay until start of animation	Here	Animation
animation-direction	Indicates direction of animation	Here	Animation
animation-duration	Indicates time an animation takes to play	Here	Animation
animation-fill-mode	Style element takes when the animation is not playing	Here	Animation
animation-iteration-count	Number of times an animation should play	Here	Animation
animation-name	Name of animation used in a @keyframes animation	Here	Animation
animation-play-state	Whether the animation is paused or not	Here	Animation
animation-timing-function	The speed curve of an animation	Here	Animation
backface-visibility	Whether element should remain visible when not facing the screen	Here	Transform
background	A shorthand property combining (up to) 8 background properties	Here	Background
background-attachment	Whether background image is fixed or scrolls with rest of page	Here	Background
background-blend-mode	The blending mode of each background layer	Here	Background
background-clip	Painting area of the background	Here	Background
background-color	Background colour of an element	Here	Background
background-image	One or more background images for element	Here	Background
background-origin	Where the background image for element is positioned	Here	Background
background-position	The (starting) position of a background image	Here	Background

background-repeat	How background image repeated	Here	Background
background-size	Size of background image(s)	Here	Background
border	A shorthand property combining (up to) 3 border properties	Here	Border
border-bottom	A shorthand property combining all main bottom border properties	Here	Border
border-bottom-color	Colour of bottom border of element	Here	Border
border-bottom-left-radius	Shape of the bottom-left corner of a border	Here	Border
border-bottom-right-radius	Shape of the bottom-right corner of a border	Here	Border
border-bottom-style	Style of bottom border of element	Here	Border
border-bottom-width	Width of bottom border of element	Here	Border
border-collapse	Whether table borders are collapsed	Here	Table
border-color	The colour of an element's four borders	Here	Border
border-image	A shorthand property combining (up to) 5 border-image properties	Here	Border
border-image-outset	Amount by which a border image area extends beyond the border box	Here	Border
border-image-repeat	How border image should be repeated, rounded or stretched	Here	Border
border-image-slice	How any border image should be sliced	Here	Border
border-image-source	Path of image to be used as a border	Here	Border
border-image-width	Width of border image	Here	Border
border-left	A shorthand property combining all main left border properties	Here	Border
border-left-color	Colour of left border of element	Here	Border
border-left-style	Style of left border of element	Here	Border
border-left-width	Width of left border of element	Here	Border
border-radius	Adds rounded corners to element	Here	Border
border-right	A shorthand property combining all main right border properties	Here	Border
border-right-color	Colour of right border of element	Here	Border
border-right-style	Style of right border of element	Here	Border
border-right-width	Width of right border of element	Here	Border
border-spacing	Distance between borders of adjacent cells (if border-collapse property is separate)	Here	Table
border-style	Style of an element's four borders	Here	Border

border-top	A shorthand property combining all main top border properties	Here	Border
border-top-color	Colour of top border of element	Here	Border
border-top-left-radius	Shape of the top -left corner of a border	Here	Border
border-top-right-radius	Shape of the top -right corner of a border	Here	Border
border-top-style	Style of top border of element	Here	Border
border-top-width	Width of top border of element	Here	Border
border-width	Width of an element's four borders	Here	Border
bottom	Bottom edge of element relative to the corresponding edge of nearest positioned ancestor	Here	Box
box-shadow	Applies one or more shadows to element	Here	Border
box-sizing	What box sizing (height, width) should be applied to	Here	User Interface
caption-side	Where a table caption should be placed	Here	Table
clear	Which sides a floating element is not allowed to float	Here	Box
clip	What to do with an image that is larger than its containing element	Here	Box
color	Colour of text within an element	Here	Colour
column-count	Number of columns element should be divided into	Here	Column Layout
column-fill	How to fill columns	Here	Column Layout
column-gap	What gap to place between columns	Here	Column Layout
column-rule	A shorthand property combining (up to) 3 column-rule properties	Here	Column Layout
column-rule-color	Colour of any rule between columns	Here	Column Layout
column-rule-style	Style of any rule between columns	Here	Column Layout
column-rule-width	Width of any rule between columns	Here	Column Layout
column-span	How many columns an element should span across	Here	Column Layout
column-width	Suggested optimal width for columns	Here	Column Layout
columns	A shorthand property for setting certain column properties	Here	Column Layout
content	Pseudo-property used with the :before and :after pseudo-elements to insert generated content	Here	User Interface
counter-increment	Pseudo-property that increments one or more CSS counter values	Here	Counters
counter-reset	Pseudo-property that creates or resets one or more CSS counter values	Here	Counters
cursor	Type of cursor to be displayed	Here	User Interface
direction	Text or writing direction	Here	Text

display	Type of box to be used for element	Here	Box
empty-cells	Whether to display borders and background for empty table cells	Here	Table
filter	Applies visual effects like grayscale, blur and saturation	Here	Images
flex	A shorthand property for setting certain flex properties	Here	Flexible Container
flex-basis	Initial length of a flexible element	Here	Flexible Container
flex-direction	Direction of a flexible element	Here	Flexible Container
flex-flow	A shorthand property for setting certain flex properties	Here	Flexible Container
flex-grow	How much an element will grow relative to other flexible items in a container	Here	Flexible Container
flex-shrink	How much an element will shrink relative other flexible items in a container	Here	Flexible Container
flex-wrap	Whether flexible items should wrap	Here	Flexible Container
float	Whether element should float	Here	Box
font	A shorthand property for setting font properties	Here	Font
@font-face	A rule allowing designers to apply their own font	Here	Font, Rules
font-family	Font to be used	Here	Font
font-size	Size of font to be used	Here	Font
font-size-adjust	Refined sizing of font to be used	Here	Font
font-stretch	Makes text in an element narrower or more stretched out than usual	Here	Font
font-style	Font style to be used	Here	Font
font-variant	Whether text should be in small-caps font	Here	Font
font-weight	How thick or thin (i.e. bold or not) characters should be	Here	Font
hanging-punctuation	Whether a punctuation mark can be placed outside box at start or end of full line of text	Here	Text
height	Height of an element (excluding padding, borders and margins)	Here	Box
justify-content	How to align a flexible container's items when the items do not use all available space along the horizontal axis	Here	Flexible Container
@keyframes	A rule allowing designers to specify animations	Here	Animation, Rules
left	Left edge of element relative to the corresponding edge of nearest positioned ancestor	Here	Box
letter-spacing	Amount of space between consecutive text characters	Here	Text
line-height	Height of lines of text	Here	Text
list-style	A shorthand property combining up to 3	Here	List

	list properties		
list-style-image	image to use as the list-item marker	Here	List
list-style-position	Whether a list marker is inside or outside the relevant content container	Here	List
list-style-type	Type of list-item marker	Here	List
margin	A shorthand property combining all four margin properties	Here	Margin
margin-bottom	Width of bottom margin of element	Here	Margin
margin-left	Width of left margin of element	Here	Margin
margin-right	Width of right margin of element	Here	Margin
margin-top	Width of top margin of element	Here	Margin
max-height	Maximum height element can become	Here	Box
max-width	Maximum width element can become	Here	Box
@media	A rule allowing designers to apply different styles for different devices and/or media types	Here	Rule
min-height	Minimum height element can become	Here	Box
min-width	Minimum width element can become	Here	Box
nav-down	Where to navigate with down arrow key	Here	User Interface
nav-index	The sequential navigation order (i.e. the 'tabbing order') for an element	Here	User Interface
nav-left	Where to navigate with left arrow key	Here	User Interface
nav-right	Where to navigate with right arrow key	Here	User Interface
nav-up	Where to navigate with up arrow key	Here	User Interface
opacity	Degree of opacity (transparency)	Here	Colour
order	Order of a flexible item relative to other flexible items inside the same container	Here	Flexible Container
orphans	Minimum number of lines of a paragraph in a paged media that can be left on an old page	Here	Paged media
outline	A shorthand property combining (up to) 3 outline properties	Here	User Interface
outline-color	Colour of outline	Here	User Interface
outline-offset	amount of space between an element's outline and edge or border of element	Here	User Interface
outline-style	Style to outline	Here	User Interface
outline-width	Width to outline	Here	User Interface
overflow	What happens when content overflows an element's box	Here	Box
overflow-x	What to with left/right edges of content overflowing an element's box	Here	Box
overflow-y	What to with top/bottom edges of content overflowing an element's box	Here	Box
padding	A shorthand property combining the 4 padding sub-properties	Here	Padding
padding-bottom	Width of the bottom padding	Here	Padding
padding-left	Width of the left padding	Here	Padding
padding-right	Width of the right padding	Here	Padding

padding-top	Width of the top padding	Here	Padding
page-break-after	Whether a page break should occur after element	Here	Page Media
page-break-before	Whether a page break should occur before element	Here	Page Media
page-break-inside	Whether a page break allowed inside element	Here	Page Media
perspective	How far 3D element is notionally placed behind the screen	Here	Transform
perspective-origin	Where 3D element is notionally placed	Here	Transform
position	How element should be positioned	Here	Box
quotes	How quotation marks should be rendered	Here	Text
resize	Whether element can be resized by user	Here	User Interface
right	Right edge of element relative to the corresponding edge of nearest positioned ancestor	Here	Box
tab-size	Size of space used for tab character	Here	
table-layout	Algorithm used to define table layout	Here	Table
text-align	How text in element should be aligned	Here	Text
text-align-last	How last line of text in element should be aligned	Here	Text
text-decoration	The 'decoration' (e.g. underlining) added to text	Here	Text Decoration
text-decoration-color	Colour of text decoration added to text	Here	Text Decoration
text-decoration-line	Line type of text decoration added to text	Here	Text Decoration
text-decoration-style	Line style of text decoration added to text	Here	Text Decoration
text-indent	Indentation applied to first line of text	Here	Text
text-justify	Type of justification applied to first line of text (if applicable)	Here	Text
text-overflow	How text that has overflowed is to be rendered by browser	Here	Text, User Interface
text-shadow	What shadow should be added to text	Here	Text Decoration
text-transform	Capitalisation to use for text	Here	Text
top	Top edge of element relative to the corresponding edge of nearest positioned ancestor	Here	Box
transform	Applies 2D or 3D transformation	Here	Transform
transform-origin	Origin used by the transform property	Here	Transform
transform-style	How nested elements are to be rendered for 3D purposes when using the transform property	Here	Transform
transition	A shorthand property combining the 4 transition sub-properties	Here	Transitions

transition-delay	When transition should start	Here	Transitions
transition-duration	How long transition will take to complete	Here	Transitions
transition-property	Properties that change as part of a transition effect	Here	Transitions
transition-timing-function	Speed curve used for a transition effect	Here	Transitions
unicode-bidi	Whether text direction should be overridden to support multiple languages	Here	Text
user-select	Whether text of element can be selected	Here	Text
vertical-align	Vertical alignment of element	Here	Box
visibility	Whether element is visible	Here	Box
white-space	How white-space inside element is handled	Here	Text
widows	Minimum number of lines of a paragraph in a paged media that can fall to a new page	Here	Paged media
width	Width of an element (excluding padding, borders and margins)	Here	Basic
word-break	Way in which words can be broken at line ends for non-CJK scripts	Here	Text
word-spacing	Amount of whitespace between words	Here	Text
word-wrap	Whether long words can be broken at line ends and wrap onto the next line	Here	Text
z-index	Stack order of an element, i.e. whether it is “in front of” other elements, and hence is visible if several would otherwise appear in the same place	Here	Box

At the time of writing there were also some other CSS properties and rules being developed by some organisations including:

- box-decoration-break
- break-after
- break-before
- break-inside
- @font-feature-values
- font-feature-settings
- font-kerning
- font-language-override
- font synthesis
- font-variant-alternatives
- font-variant-caps
- font-variant-east-asian
- font-variant-ligatures
- font-variant-numeric
- font-variant-position
- hyphens
- icon (is not currently supported by many major browsers)
- image-orientation

- image-rendering
- image-resolution
- ime-mode
- line-break
- mark
- mark-after
- mark-before
- marks
- marquee-direction
- marquee-play-count
- marquee-speed
- marquee-style
- mask
- mask-type
- object-fit
- object-position
- orphans
- overflow-wrap
- phonemes
- rest
- rest-after
- rest-before
- text-combine-upright
- text-orientation
- text-underline-position
- voice-balance
- voice-duration
- voice-pitch
- voice-pitch-range
- voice-rate
- voice-stress
- voice-volume
- widows
- writing-mode

Individual CSS Properties:

align-content

[CSSPropertyAlignContent]

The CSS (CSS3) `align-content` property modifies the behaviour of the flex-wrap property, aligning flex lines. N.B. If you want to align items on the main *x*-axis then use the justify-content property.

Valid property values (other than inherit and initial) are:

Value	Description
`center`	Lines packed towards centre of flex container
`flex-end`	Lines packed towards end of flex container
`flex-start`	Lines packed towards start of flex container
`space-between`	Lines distributed evenly in flex container
`space-around`	Lines distributed evenly in flex container but with half-size spaces at either end
`stretch`	(default value). Lines stretch to take up remaining space

Default Value: `stretch`
JavaScript syntax: e.g. *object*`.style.alignContent = "center";`
Inherited: No
Animatable: No

align-items

[CSSPropertyAlignItems]

The CSS (CSS3) `align-items` property specifies the default alignment for items inside a flexible container. Use the align-self property to override the property for any individual item.

Valid property values (other than inherit and initial) are:

Value	Description
`baseline`	Items positioned at baseline of container
`center`	Items positioned at centre of container
`flex-end`	Items positioned at end of container
`flex-start`	Items positioned at start of container
`stretch`	(default value). Items stretched to fit container

Default Value: `stretch`
JavaScript syntax: e.g. *object*`.style.alignItems = "center";`
Inherited: No
Animatable: No

align-self

[CSSPropertyAlignSelf]

The CSS (CSS3) `align-self` property specifies the alignment for the selected item within a flexible container. Use the align-items property to set the default that otherwise applies to the items in the flexible container.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Element inherits its parent container's align-items property, or "stretch" if it has no parent container
`baseline`	Items positioned at baseline of container
`center`	Items positioned at centre of container
`flex-end`	Items positioned at end of container
`flex-start`	Items positioned at start of container
`stretch`	Items stretched to fit container

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.alignSelf = "center";`
Inherited: No
Animatable: No

all

[CSSPropertyAll]

The CSS (CSS3) `all` property resets all properties, apart from unicode-bidi and direction, to their initial or inherited values.

Valid property values (other than inherit and initial) are:

Value	Description
`unset`	Changes all properties applied to element or its parent to their parent value if they are inheritable or to their initial value if not

Default Value: N/A
JavaScript syntax: e.g. *object*`.style.all = "initial";`
Inherited: No
Animatable: No

background

[CSSPropertyBackground]

The CSS (CSS1 and CSS3) `background` property is a shorthand property combining (up to) 8 of the background properties.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- background-color

- background-image
- background-position
- background-size
- background-repeat
- background-origin
- background-clip
- background-attachment

Default Value:	*See individual properties*
JavaScript syntax:	e.g. *object*`.style.background="red url(mypicture.gif) top left no-repeat"`
Inherited:	No
Animatable**:**	See *individual properties*

If one of the properties in the shorthand declaration is the background-size property then you need to use a "/" to separate it from the background-position property (the background-size property was added in CSS3), e.g.:

```
div {background: url(mypicture.gif) 10px 20px/40px 40px;}
```

More than one background-image can be specified. However, if you are using multiple background-image sources and also want a background-color then you need to put the background-color parameter last in the list.

background-attachment

[CSSPropertyBackgroundAttachment]

The CSS (CSS1) `background-attachment` property indicates whether a background image is fixed or scrolls with the rest of the page.

Valid property values (other than inherit and initial) are:

Value	Description
`fixed`	Background is fixed in relation to viewport
`local`	Background scrolls along with the element's contents
`scroll`	(default value). Background scrolls along with the element

Default Value: `scroll`
JavaScript syntax: e.g. *object*`.style.backgroundAttachment="fixed"`
Inherited: No
Animatable: No

background-blend-mode

[CSSPropertyBackgroundBlendMode]

The CSS (CSS3) `background-blend-mode` property defines the blending mode of each background layer (i.e. colour and/or image).

Valid property values are:

Value	Description
`color`	Blending mode set to color
`color-dodge`	Blending mode set to color-dodge
`darken`	Blending mode set to darken
`lighten`	Blending mode set to lighten
`luminosity`	Blending mode set to luminosity
`multiply`	Blending mode set to multiply
`normal`	(default value). Blending mode set to normal
`overlay`	Blending mode set to overlay
`saturation`	Blending mode set to saturation
`screen`	Blending mode set to screen

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.backgroundBlendMode="screen"`
Inherited: No
Animatable: No

background-clip

[CSSPropertyBackgroundClip]

The CSS (CSS3) `background-clip` property specifies the painting area of the background.

Valid property values (other than inherit and initial) are:

Value	Description
`border-box`	(default value). Background is clipped to the border box
`content-box`	Background is clipped to the content box
`padding-box`	Background is clipped to the padding box

Default Value: `border-box`
JavaScript syntax: e.g. *object*`.style.backgroundClip="content-box"`
Inherited: No
Animatable: No

background-color

[CSSPropertyBackgroundColor]

The CSS (CSS1) `background-color` property sets the background colour of an element. The background of an element includes its padding and border but *not* its margin. Usually, a background colour and text colour should be chosen in tandem to make the text easy to read.

Valid property values (other than inherit and initial) are:

Value	Description
color	Specified CSS colour
`transparent`	(default value). Transparent

Default Value: `transparent`
JavaScript syntax: e.g. *object*`.style.backgroundColor="red"`
Inherited: No
Animatable**:** Yes

background-image

[CSSPropertyBackgroundImage]

The CSS (CSS1) `background-image` property sets one or more background images for an element. The background of an element includes its padding and border but *not* its margin. Usually, you should set a background-color to be used if the image is unavailable and the background image and text colour should be chosen in tandem to make the text easy to read.

By default, a background image is placed at the top-left corner of an element and is repeated both vertically and horizontally. These features can be overridden using the background-position and background-repeat properties.

Valid property values (other than inherit and initial) are:

Value	Description
`url(imagefile)`	The URL of the image. To specify more than one image, separate the URLs with a comma
`none`	(default value). No background image supplied

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.backgroundImage="url(mypicture.gif)"`
Inherited: No
Animatable**:** No

background-origin

[CSSPropertyBackgroundOrigin]

The CSS (CSS3) `background-origin` property specifies where the background image for an element is positioned. If the background-attachment property is set to "fixed" then the background-origin property has no effect.

Valid property values (other than inherit and initial) are:

Value	Description
`border-box`	Background image starts from the upper-left corner of the border
`content-box`	Background image starts from the upper-left corner of the content
`padding-box`	(default value). Background image starts from the upper-left corner of the padding edge

Default Value: `padding-box`
JavaScript syntax: e.g. *object*`.style.backgroundOrigin="content-box"`
Inherited: No
Animatable**:** No

background-position

[CSSPropertyBackgroundPosition]

The CSS (CSS1) `background-position` property sets the (starting) position of a background image. By default, a background-image is placed at the top-left corner of an element and is repeated both vertically and horizontally.

Valid property values (other than inherit and initial) are:

Value	Description
value	See below
x% y%	Horizontal position and vertical position as fraction of size of element. Top left is `0% 0%`, bottom right is `100% 100%`. If you only specify one value then the other will be `50%`.
xpos ypos	(e.g. 0px 0px). Horizontal position and vertical position in any valid CSS length. You can mix % and positions

Default Value: `0% 0%`
JavaScript syntax: e.g. *object*`.style.backgroundPosition="10% 20%"`
Inherited: No
Animatable**:** Yes

Other acceptable values include combinations of `left`, `center`, `right` (for *xpos*) and `top`, `center`, `bottom` (for *ypos*). If you only specify one keyword then the other value will be `center`.

background-repeat

[CSSPropertyBackgroundRepeat]

The CSS (CSS1) `background-repeat` property sets whether/how a background image will be repeated. By default, a background-image is placed at the top-left corner of an element and is repeated both vertically and horizontally.

Valid property values (other than inherit and initial) are:

Value	Description
`no-repeat`	Not repeated
`repeat`	(default value). Repeated both vertically and horizontally
`repeat-x`	Repeated only horizontally
`repeat-y`	Repeated only vertically

Default Value: `repeat`
JavaScript syntax: e.g. *object*`.style.backgroundRepeat="repeat-x"`
Inherited: No
Animatable**:** No

background-size

[CSSPropertyBackgroundSize]

The CSS (CSS3) `background-size` property specifies the size of the background image(s).

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Background-image contains its width and height
`contain`	Scales the image to be as large as possible such that both width and height fit inside the content area
`cover`	Scales the image to be as large as possible to cover the background area completely
length	Sets width and height in that order using any valid CSS length. If only one is given the second is set to `auto`
percentage	Sets width and height in that order as percentages of the parent element. If only one is given the second is set to `auto`

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.backgroundSize="60px 80px"`
Inherited: No
Animatable**:** Yes

`

border

[CSSPropertyBorder]

The CSS (CSS1) `border` property is a shorthand property combining (up to) 3 of the border properties.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- border-width
- border-style

- border-color

Default Value: *See individual properties*
JavaScript syntax: e.g. *object*`.style.border="2px solid red"`
Inherited: No
Animatable**:** See *individual properties*

border-bottom

[CSSPropertyBorderBottom]

The CSS (CSS1) `border-bottom` property is a shorthand property combining all the main bottom border properties.

Valid property values (other than inherit or initial) are defined by the elements of the shorthand and are:

- border-bottom-width
- border-bottom-style
- border-bottom-color

Missing properties are given their default values.

Default Value: `medium none` *color*
JavaScript syntax: e.g. *object*`.style.borderBottom="2px solid red"`
Inherited: No
Animatable**:** Yes

In the default value, *color* is the CSS colour of the element.

border-bottom-color

[CSSPropertyBorderBottomColor]

The CSS (CSS1) `border-bottom-color` property sets the colour of the bottom border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
color	Specified CSS colour
`transparent`	Transparent

Default Value: The current CSS colour of the element
JavaScript syntax: e.g. *object*.style`.borderBottomColor="red"`
Inherited: No
Animatable**:** Yes

border-bottom-left-radius

[CSSPropertyBorderBottomLeftRadius]

The CSS (CSS3) `border-bottom-left-radius` property determines the shape of the bottom-left corner of a border. It allows you to add rounded borders to elements.

The two length or percentage values define the radii of a quarter ellipse defining the shape of the corner. The first is the horizontal radius, the second the vertical radius. If either is omitted then it is copied from the other (so the corner is a quarter-circle). If either is zero then the corner becomes square. Percentages for the horizontal radius refer to the width of the border box, those for the vertical radius refer to the height of the border box.

Valid property values (other than inherit and initial) are:

Value	Description
length1, length2	Specified CSS lengths
%, %	See above

Default Value: 0
JavaScript syntax: e.g. *object*`.style.borderBottomLeftRadius="5px 8px"`
Inherited: No
Animatable**:** Yes

border-bottom-right-radius

[CSSPropertyBorderBottomRightRadius]

The CSS (CSS3) `border-bottom-right-radius` property determines the shape of the bottom-right corner of a border. It allows you to add rounded borders to elements.

The two length or percentage values define the radii of a quarter ellipse defining the shape of the corner. The first is the horizontal radius, the second the vertical radius. If either is omitted then it is copied from the other (so the corner is a quarter-circle). If either is zero then the corner becomes square. Percentages for the horizontal radius refer to the width of the border box, those for the vertical radius refer to the height of the border box.

Valid property values (other than inherit and initial) are:

Value	Description
length1, length2	Specified CSS lengths
%, %	See above

Default Value: 0
JavaScript syntax: e.g. *object*`.style.borderBottomRightRadius="5px 8px"`
Inherited: No
Animatable**:** Yes

border-bottom-style

[CSSPropertyBorderBottomStyle]

The CSS (CSS1) `border-bottom-style` property sets the border style of the bottom border of an element.

Valid property values (other than inherit and initial) are shown here.

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.borderBottomStyle="groove"`
Inherited: No
Animatable: No

border-bottom-width

[CSSPropertyBorderBottomWidth]

The CSS (CSS1) `border-bottom-width` property sets the width of the bottom border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
`medium`	(default value). Medium width
`thick`	Thick width
`thin`	Thin width

Default Value: `medium`
JavaScript syntax: e.g. *object*`.style.borderBottomWidth="4px"`
Inherited: No
Animatable: Yes

border-collapse

[CSSPropertyBorderCollapse]

The CSS (CSS2) `border-collapse` property indicates whether table borders are collapsed into a single border or detached as in standard HTML. Note if a !DOCTYPE is not specified then the border-collapse property can produce unexpected results.

Valid property values (other than inherit and initial) are:

Value	Description
`collapse`	Borders are collapsed into a single border where possible (ignoring border-spacing and empty-cells properties)
`separate`	(default value). Borders are detached, and border-spacing and empty-cells properties have some meaning

Default Value: `separate`
JavaScript syntax: e.g. *object*`.style.borderCollapse="collapse"`

Inherited: Yes
Animatable: No

border-color

[CSSPropertyBorderColor]

The CSS (CSS1) `border-color` property sets the colour of an element's four borders. The individual border colours can be set separately using border-bottom-color, border-left-color, border-right-color and border-top-color. As with some other aggregate edge properties, up to four parameter values can be supplied (and if more than one is supplied then the properties are applied to individual borders as described here).

Valid property values (other than inherit and initial) are:

Value	Description
color	A CSS colour
`transparent`	(default value). Transparent

Default Value: The current CSS colour of the element
JavaScript syntax: e.g. *object*`.style.borderColor="red blue"`
Inherited: No
Animatable: Yes

border-image

[CSSPropertyBorderImage]

The CSS (CSS1) `border-image` property is a shorthand property combining (up to) 5 of the border-image properties. These allow you to specify that an image should be used instead of the normal border around an element.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- border-image-source
- border-image-slice
- border-image-width
- border-image-outset
- border-image-repeat

Default Value: `none 100% 1 0 stretch`
JavaScript syntax: e.g. *object*`.style.borderImage="url(myborder.png) 20 round"`
Inherited: No
Animatable: No

border-image-outset

[CSSPropertyBorderImageOutset]

The CSS (CSS3) `border-image-outset` property specifies the amount by which a border image area extends beyond the border box.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length that specifies how far from the edges the border-image will appear
number	Multiples of the relevant border-width

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.borderImageOutset="30px"`
Inherited: No
Animatable: No

border-image-repeat

[CSSPropertyBorderImageRepeat]

The CSS (CSS3) `border-image-repeat` property specifies whether and how the border image should be repeated, rounded or stretched.

Valid property values (other than inherit and initial) are:

Value	Description
`repeat`	Image tiled (i.e. repeated) to fill area
`round`	Image tiled (i.e. repeated) to fill area (with image rescaled so that it exactly fits the area)
`space`	Image tiled (i.e. repeated) to fill area (with extra space distributed around tiles if the fit is not exact
`stretch`	(default value). Image stretch to fill area

Default Value: `stretch`
JavaScript syntax: e.g. *object*`.style.borderImageRepeat="space"`
Inherited: No
Animatable: No

border-image-slice

[CSSPropertyBorderImageSlice]

The CSS (CSS3) `border-image-slice` property specifies how any border image should be sliced. The image is always sliced into nine sections (3 x 3, i.e. 4 corners, 4 edge non-corners and 1 middle). The middle part is fully transparent unless the fill keyword is set.

The property takes up to four values. If the fourth is omitted then it is given the same value as the second. If the third is omitted then it is given the same value as the first. If the second is also omitted then it is given the same value as the first.

Valid property values (other than inherit and initial) are:

Value	Description
number	Number(s) of pixels for raster images or coordinates for vector images
x%	Percentages relative to height or width of image
`fill`	Middle part of image is displayed

Default Value: `100%`
JavaScript syntax: e.g. *object*`.style.borderImageSlice="20%"`
Inherited: No
Animatable: No

border-image-source

[CSSPropertyBorderImageSource]

The CSS (CSS3) `border-image-source` property specifies path of image to be used as a border (instead of a normal border around an element).

Valid property values (other than inherit and initial) are:

Value	Description
`none`	(default value). No image used
url('URL')	URL path to the border image

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.borderImageSource="url(myborder.png)"`
Inherited: No
Animatable: No

border-image-width

[CSSPropertyBorderImageWidth]

The CSS (CSS3) `border-image-width` property specifies the width of a border image.

The property takes up to four values. If the fourth is omitted then it is given the same value as the second. If the third is omitted then it is given the same value as the first. If the second is also omitted then it is given the same value as the first.

Valid property values (other than inherit and initial) are:

Value	Description
length	CSS length specifying size of border-width
number	(default value). Multiples of corresponding border-width
%	Relative to size of border image area
`auto`	Intrinsic width or height of the corresponding image slice

Default Value: 1
JavaScript syntax: e.g. *object*`.style.borderImageWidth="50px"`
Inherited: No
Animatable**:** No

border-left

[CSSPropertyBorderLeft]

The CSS (CSS1) `border-left` property is a shorthand property combining all the main left border properties.

Valid property values (other than inherit or initial) are defined by the elements of the shorthand and are:

- border-left-width
- border-left-style
- border-left-color

Missing properties are given their default values.

Default Value: `medium none` *color*
JavaScript syntax: e.g. *object*`.style.borderLeft="2px solid red"`
Inherited: No
Animatable**:** Yes

In the default value, *color* is the CSS colour of the element.

border-left-color

[CSSPropertyBorderLeftColor]

The CSS (CSS1) `border-left-color` property sets the colour of the left border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
color	Specified CSS colour
`transparent`	Transparent

Default Value: The current CSS colour of the element
JavaScript syntax: e.g. *object*`.style.borderLeftColor="red"`
Inherited: No
Animatable**:** Yes

border-left-style

[CSSPropertyBorderLeftStyle]

The CSS (CSS1) `border-left-style` property sets the border style of the left border of an element.

Valid property values (other than inherit and initial) are shown here.

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.borderLeftStyle="groove"`
Inherited: No
Animatable: No

border-left-width

[CSSPropertyBorderLeftWidth]

The CSS (CSS1) `border-left-width` property sets the width of the left border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
`medium`	(default value). Medium width
`thick`	Thick width
`thin`	Thin width

Default Value: `medium`
JavaScript syntax: e.g. *object*`.style.borderLeftWidth="4px"`
Inherited: No
Animatable: Yes

border-radius

[CSSPropertyBorderRadius]

The CSS (CSS1) `border-radius` property is used to add rounded corners to an element. It is a shorthand way of setting the four individual border radius properties, see:

- border-top-left-radius
- border-top-right-radius
- border-bottom-right-radius
- border-bottom-left-radius

Valid property values (other than inherit and initial) are (single-value versions) of the valid properties for each of these individual elements.

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.borderRadius="4px"`
Inherited: No
Animatable: Yes

Depending on the number of values supplied:

Number supplied	E.g.	Which values are applied to which corners
1	*x1*	*x1* applied to all four corners
2	*x1 x2*	*x1* applied to top-left and bottom-right corners *x2* applied to top-right and bottom-left corners
3	*x1 x2 x3*	*x1* applied to top-left corner *x2* applied to top-right and bottom-left corners *x3* applied to bottom-right corner
4	*x1 x2 x3 x4*	*x1* applied to top-left corner *x2* applied to top-right corner *x3* applied to bottom-right corner *x4* applied to bottom-left corner

border-right

[CSSPropertyBorderRight]

The CSS (CSS1) `border-right` property is a shorthand property combining all the main right border properties.

Valid property values (other than inherit or initial) are defined by the elements of the shorthand and are:

- border-right-width
- border-right-style
- border-right-color

Missing properties are given their default values.

Default Value: `medium none` *color*
JavaScript syntax: e.g. *object*`.style.borderRight="2px solid red"`
Inherited: No
Animatable**:** Yes

In the default value, *color* is the CSS colour of the element.

border-right-color

[CSSPropertyBorderRightColor]

The CSS (CSS1) `border-right-color` property sets the colour of the right border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
color	Specified CSS colour

`transparent`	Transparent

Default Value: The current CSS colour of the element
JavaScript syntax: e.g. *object*`.style.borderRightColor="red"`
Inherited: No
Animatable**:** Yes

border-right-style

[CSSPropertyBorderRightStyle]

The CSS (CSS1) `border-right-style` property sets the border style of the right border of an element.

Valid property values (other than inherit and initial) are shown here.

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.borderRightStyle="groove"`
Inherited: No
Animatable**:** No

border-right-width

[CSSPropertyBorderRightWidth]

The CSS (CSS1) `border-right-width` property sets the width of the right border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
`medium`	(default value). Medium width
`thick`	Thick width
`thin`	Thin width

Default Value: `medium`
JavaScript syntax: e.g. *object*`.style.borderRightWidth="4px"`
Inherited: No
Animatable**:** Yes

border-spacing

[CSSPropertyBorderSpacing]

The CSS (CSS2) `border-spacing` property sets the distance between borders of adjacent cells (if the border-collapse property is `separate` (which is its default).

Valid property values (other than inherit and initial) are:

Value	Description
length1 length2	Distance between borders of adjacent cells in CSS lengths. Must be non-negative. If one value supplied then both horizontal and vertical spacing. If two supplied then first relates to horizontal spacing and second to vertical spacing
`medium`	(default value). Medium width
`thick`	Thick width
`thin`	Thin width

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.borderSpacing="4px"`
Inherited: No
Animatable**:** Yes

border-style

[CSSPropertyBorderStyle]

The CSS (CSS1) `border-style` property sets the border style of an element's four borders. The individual border styles can be set separately using border-bottom-style, border-left-style, border-right-style and border-top-style. As with some other aggregate edge properties, up to four parameter values can be supplied (and if more than one is supplied then the properties are applied to individual borders as described here).

Valid property values (other than inherit and initial) are shown here.

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.borderStyle="double dashed"`
Inherited: No
Animatable**:** Yes

border-top

[CSSPropertyBorderTop]

The CSS (CSS1) `border-top` property is a shorthand property combining all the main top border properties.

Valid property values (other than inherit or initial) are defined by the elements of the shorthand and are:

- border-top-width
- border-top-style
- border-top-color

Missing properties are given their default values.

Default Value: `medium none` *color*
JavaScript syntax: e.g. *object*`.style.borderTop="2px solid red"`
Inherited: No

Animatable: Yes

In the default value, *color* is the CSS colour of the element.

border-top-color

[CSSPropertyBorderTopColor]

The CSS (CSS1) `border-top-color` property sets the colour of the top border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
color	Specified CSS colour
`transparent`	Transparent

Default Value: The current CSS colour of the element
JavaScript syntax: e.g. *object*`.style.borderTopColor="red"`
Inherited: No
Animatable: Yes

border-top-left-radius

[CSSPropertyBorderTopLeftRadius]

The CSS (CSS3) `border-top-left-radius` property determines the shape of the top-left corner of a border. It allows you to add rounded borders to elements.

The two length or percentage values define the radii of a quarter ellipse defining the shape of the corner. The first is the horizontal radius, the second the vertical radius. If either is omitted then it is copied from the other (so the corner is a quarter-circle). If either is zero then the corner becomes square. Percentages for the horizontal radius refer to the width of the border box, those for the vertical radius refer to the height of the border box.

Valid property values (other than inherit and initial) are:

Value	Description
length1, length2	Specified CSS lengths
%, %	See above

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.borderTopLeftRadius="5px 8px"`
Inherited: No
Animatable: Yes

border-top-right-radius

[CSSPropertyBorderTopRightRadius]

The CSS (CSS3) `border-top-right-radius` property determines the shape of the top-right corner of a border. It allows you to add rounded borders to elements.

The two length or percentage values define the radii of a quarter ellipse defining the shape of the corner. The first is the horizontal radius, the second the vertical radius. If either is omitted then it is copied from the other (so the corner is a quarter-circle). If either is zero then the corner becomes square. Percentages for the horizontal radius refer to the width of the border box, those for the vertical radius refer to the height of the border box.

Valid property values (other than inherit and initial) are:

Value	Description
length1, length2	Specified CSS lengths
%, %	See above

Default Value: *0*
JavaScript syntax: e.g. *object*`.style.borderTopRightRadius="5px 8px"`
Inherited: No
Animatable**:** Yes

border-top-style

[CSSPropertyBorderTopStyle]

The CSS (CSS1) `border-top-style` property sets the border style of the top border of an element.

Valid property values (other than inherit and initial) are shown here.

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.borderTopStyle="groove"`
Inherited: No
Animatable**:** No

border-top-width

[CSSPropertyBorderTopWidth]

The CSS (CSS1) `border-top-width` property sets the width of the top border of an element. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
`medium`	(default value). Medium width
`thick`	Thick width
`thin`	Thin width

Default Value: `medium`
JavaScript syntax: e.g. *object*`.style.borderTopWidth="4px"`
Inherited: No
Animatable: Yes

border-width

[CSSPropertyBorderWidth]

The CSS (CSS1) `border-width` property sets the width of an element's four borders. You should always specify the border-style property before this property, as an element must have a border before you can change its characteristics. The individual border widths can be set separately using border-bottom-width, border-left-width, border-right-width and border-top-width. As with some other aggregate edge properties, up to four parameter values can be supplied (and if more than one is supplied then the properties are applied to individual borders as described here).

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
`medium`	(default value). Medium width
`thick`	Thick width
`thin`	Thin width

Default Value: `medium`
JavaScript syntax: e.g. *object*`.style.borderWidth="10px 10px"`
Inherited: No
Animatable: Yes

bottom

[CSSPropertyBottom]

The CSS (CSS2) `bottom` property sets, for absolutely positioned elements, the bottom edge of an element relative to the corresponding edge of its nearest positioned ancestor. If such an element has no positioned ancestors then it uses the document body and moves along with page scrolling. A 'positioned' element is one whose position is anything other than static.

Valid property values (other than inherit and initial) are:

Value	Description
length	Sets edge position in CSS length units. Can be negative
%	Sets edge position in % of containing element. Can be negative
`auto`	(default value). Browser calculates edge position

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.bottom="10px"`
Inherited: No
Animatable: Yes

box-shadow

[CSSPropertyBoxShadow]

The CSS (CSS3) `box-shadow` property applies one or more shadows to an element. The property is a comma-separated list of shadows, each specified by 2 to 4 length values, an optional colour and an optional inset keyword. If a length is omitted then it is taken to be zero.

Valid property values (other than inherit and initial) are:

Value	Description
box-shadow parameter	See below
`none`	(default value). No shadow is displayed

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.boxShadow="5px 10px 15px red inset"`
Inherited: No
Animatable: Yes

The parameters of the box-shadow parameter are:

Value	Description
h-shadow	Position of horizontal shadow (can be negative)
v-shadow	Position of vertical shadow (can be negative)
blur	Optional. Blur distance
spread	Optional. Size of shadow (can be negative)
color	Optional (for most browsers). Shadow CSS colour. Default value is black.
`inset`	Optional. Changes shadow from an outer shadow to an inner shadow

box-sizing

[CSSPropertyBoxSizing]

The CSS (CSS3) `box-sizing` property indicates what the sizing properties of a box (width and height) should be applied to, i.e. just the content-box or some broader definition.

Valid property values (other than inherit and initial) are:

Value	Description
`border-box`	Height and width properties (and min-height / min-width / max-height / max-width properties) include content, border and padding (but not margin)
`content-box`	(default value). Height and width properties (and min-height / min-width / max-height / max-width properties) include only the content, and do not include border, padding or margin

Default Value: `content-box`

JavaScript syntax: e.g. *object*`.style.boxSizing="border-box"`
Inherited: No
Animatable: No

caption-side

[CSSPropertyCaptionSide]

The CSS (CSS2) `caption-side` property indicates where a table caption should be placed.

Valid property values (other than inherit and initial) are:

Value	Description
`bottom`	Puts caption below table
`top`	(default value). Puts caption above table

Default Value: `top`
JavaScript syntax: e.g. *object*`.style.captionSide="bottom"`
Inherited: Yes
Animatable: No

clear

[CSSPropertyClear]

The CSS (CSS1) `clear` property indicates on which sides of an element a floating element is not allowed to float.

Valid property values (other than inherit and initial) are:

Value	Description
`both`	Floating elements not allowed on either left or right side
`left`	Floating elements not allowed on left side
`none`	(default value). Allows floating elements on both sides
`right`	Floating elements not allowed on right side

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.clear="both"`
Inherited: No
Animatable: No

clip

[CSSPropertyClip]

The CSS (CSS2) `clip` property indicates what to do with an image that is larger than its containing element. The clip property allows you to specify a rectangle to clip an absolutely positioned element. The rectangle is specified by four coordinates, all from the top-left corner of the element being clipped. The *clip* property does not work if the overflow property is set to visible.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). No clipping is applied
shape	Clips an element to within a given shape

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.clip="rect(0px,10px,10px,5px)"`
Inherited: No
Animatable: Yes

At the time of writing, the only valid type of value for *shape* was `rect`(*top*, *right*, *bottom*, *left*) where *top*, *right*, *bottom* and *left* are CSS lengths.

color

[CSSPropertyColor]

The CSS (CSS1) `color` property indicates the colour of text within an element. Usually you should choose a combination of background colour and text colour that makes the text easy to read.

Valid property values (other than inherit and initial) are:

Value	Description
color	A CSS colour

Default Value: N/A
JavaScript syntax: e.g. *object*`.style.color="red"`
Inherited: Yes
Animatable: Yes

column-count

[CSSPropertyColumnCount]

The CSS (CSS3) `column-count` property specifies the number of columns an element (e.g. a paragraph) should be divided into.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Number of columns will be determined by other properties such as column-width
number	Optimum number of columns into which content of element will be formatted

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.columnCount=2`
Inherited: No
Animatable: Yes

column-fill

[CSSPropertyColumnFill]

The CSS (CSS3) `column-fill` property specifies how to fill columns.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	Columns filled sequentially and therefore may have different lengths
`balance`	(default value). Columns are balanced and browsers should minimise variation in column length

Default Value: `balance`
JavaScript syntax: e.g. *object*`.style.columnFill="auto"`
Inherited: No
Animatable**:** No

column-gap

[CSSPropertyColumnGap]

The CSS (CSS3) `column-gap` property specifies what gap is placed between columns of an element. Any column rule between columns will appear in the middle of the gap.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length
`normal`	(default value). Specifies normal gap between columns (W3C suggests 1em)

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.columnGap="20px"`
Inherited: No
Animatable**:** Yes

column-rule

[CSSPropertyColumnRule]

The CSS (CSS1) `column-rule` property is a shorthand property combining (up to) 3 of the column-rule properties.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- column-rule-width
- column-rule-style

- column-rule-color

Default Value: *See individual properties*
JavaScript syntax: e.g. *object*`.style.columnRule="3px outset red"`
Inherited: No
Animatable**:** *See individual properties*

column-rule-color

[CSSPropertyColumnRuleColor]

The CSS (CSS3) `column-rule-color` property specifies the colour of any rule between columns.

Valid property values (other than inherit and initial) are:

Value	Description
color	A CSS colour

Default Value: *Current colour of element*
JavaScript syntax: e.g. *object*`.style.columnRuleColor="red"`
Inherited: No
Animatable**:** Yes

column-rule-style

[CSSPropertyColumnRuleStyle]

The CSS (CSS3) `column-rule-style` property specifies the style of any rule between columns.

Valid property values (other than inherit and initial) are:

Value	Description
`dashed`	Dashed border
`dotted`	Dotted border
`double`	Double border
`groove`	Effect depends on column-rule-width and column-rule-color values
`hidden`	Hidden
`inset`	Effect depends on column-rule-width and column-rule-color values
`none`	(default value). No border
`outset`	Effect depends on column-rule-width and column-rule-color values
`ridge`	Effect depends on column-rule-width and column-rule-color values
`solid`	Solid border

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.columnRuleStyle="dotted"`
Inherited: No
Animatable**:** No

column-rule-width

[CSSPropertyColumnRuleWidth]

The CSS (CSS3) `column-rule-width` property specifies the width of any rule between columns.

Valid property values (other than inherit and initial) are:

Value	Description
length	Specified thickness as a CSS length
`medium`	(default value). Medium width
`thick`	Thick width
`thin`	Thin width

Default Value: `medium`
JavaScript syntax: e.g. *object*`.style.columnRuleWidth="5px"`
Inherited: No
Animatable**:** Yes

columns

[CSSPropertyColumns]

The CSS (CSS1) `columns` property is a shorthand property for setting certain column properties for an element.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- column-width
- column-count

Default Value: `auto auto`
JavaScript syntax: e.g. *object*`.style.columns="80px 3"`
Inherited: No
Animatable**:** *See individual properties*

column-span

[CSSPropertyColumnSpan]

The CSS (CSS3) `column-span` property specifies how many columns an element should span across.

Valid property values (other than inherit and initial) are:

Value	Description
`1`	(default value). Element should span across one column
`all`	Element should span across all columns

Default Value: `1`
JavaScript syntax: e.g. *object*`.style.columnSpan="all"`

Inherited: No
Animatable: No

column-width

[CSSPropertyColumnWidth]

The CSS (CSS3) `column-width` property provides a suggested optimal width for columns.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length
`auto`	(default value). Column width will be determined by browser

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.columnWidth="200px"`
Inherited: No
Animatable: Yes

cursor

[CSSPropertyCursor]

The CSS (CSS2) `cursor` property indicates the type of cursor to be displayed when pointing to an element. Usually this property is set in a manner that conventionally indicates what is likely to happen next (if the cursor is clicked).

Valid property values (other than inherit and initial) are:

Value	Description
`alias`	Indicates alias is to be created
`all-scroll`	Indicates that something can be scrolled in any direction
`auto`	(default value). Browser selects cursor format
`cell`	Indicates a cell (or set of cells) can be selected
`context-menu`	Indicates a context-menu is available
`col-resize`	Indicates column can be resized horizontally
`copy`	Indicates something to be copied
`crosshair`	Cursor appears as a crosshair
`default`	Cursor appears as the default cursor
`e-resize`	Indicates edge of box can be moved right (i.e. 'east')
`ew-resize`	Indicates a bidirectional resize cursor
`grab`	Indicates something can be grabbed
`grabbing`	Indicates something can be grabbed
`help`	Indicates help is available
`move`	Indicates can move something
`n-resize`	Indicates edge of box can be moved up (i.e. 'north')
`ne-resize`	Indicates edge of a box can be moved up and right (i.e. 'north-east')
`nesw-resize`	Indicates a bidirectional resize cursor
`ns-resize`	Indicates a bidirectional resize cursor

nw-resize	Indicates edge of a box can be moved up and left (i.e. 'north-west')
nwse-resize	Indicates a bidirectional resize cursor
no-drop	Indicates can't drop dragged item here
none	No cursor shown
not-allowed	Indicates requested action won't be executed
pointer	Indicates a link
progress	Indicates program is busy
row-resize	Indicates row can be resized vertically
s-resize	Indicates edge of box can be moved down (i.e. 'south')
se-resize	Indicates edge of a box can be moved down and right (i.e. 'south-east')
sw-resize	Indicates edge of a box can be moved down and left (i.e. 'south-west')
text	Indicates text may be selected
URL	A comma separated list of URLs pointing to custom cursors. You should ideally specify a generic cursor at the end of the list (in case none of the URL-defined cursors can be used)
vertical-text	Indicates vertical-text can be selected
w-resize	Indicates edge of box can be moved left (i.e. 'west')
wait	Indicates program is busy
zoom-in	Indicates something can be zoomed in
zoom-out	Indicates something can be zoomed out

Default Value: auto
JavaScript syntax: e.g. *object*.style.cursor="zoom-in"
Inherited: Yes
Animatable: No

direction

[CSSPropertyDirection]

The CSS (CSS2) direction property specifies the text or writing direction. It can be used with the unicode-bidi property to set or indicate whether text should be overridden to support multiple languages (that are formatted in different directions) in the same document.

Valid property values (other than inherit and initial) are:

Value	Description
ltr	(default value). Text direction is left-to-right
rtl	Text direction is right-to-left
zoom-in	Indicates something can be zoomed in
zoom-out	Indicates something can be zoomed out

Default Value: ltr
JavaScript syntax: e.g. *object*.style.direction="rtl"
Inherited: Yes
Animatable: No

display

[CSSPropertyDisplay]

The CSS (CSS1 and some values that are new in CSS3) `display` property indicates the type of box to be used for an element.

Valid property values (other than inherit and initial) are:

Value	Description
`block`	Displayed as a block element (like default <p> element)
`flex`	Displayed as a block-level flex container
`inline`	(default value). Displayed in-line (like default element)
`inline-block`	Displayed as an in-line block container (i.e. the inside is formatted like a block-level box but element itself as an inline-level box)
`inline-flex`	Displayed as an inline-level flex container
`inline-table`	Displayed as an inline-level table
`list-item`	Displayed like a default element
`none`	Not displayed
`run-in`	Displayed as either block or inline depending on context
`table`	Displayed like a default <table> element
`table-caption`	Displayed like a default <caption> element
`table-cell`	Displayed like a default <td> element
`table-column`	Displayed like a default <col> element
`table-column-group`	Displayed like a default <colgroup> element
`table-footer-group`	Displayed like a default <tfoot> element
`table-header-group`	Displayed like a default <thead> element
`table-row`	Displayed like a default <tr> element
`table-row-group`	Displayed like a default <tbody> element

Default Value: `inline`
JavaScript syntax: e.g. *object*`.style.display="none"`
Inherited: No
Animatable: No

empty-cells

[CSSPropertyEmptyCells]

The CSS (CSS2) `empty-cells` property indicates whether to display borders and background for empty cells in a table (only applicable if border-collapse property is "separate").

Valid property values (other than inherit and initial) are:

Value	Description
`hide`	No background or borders shown on empty cells
`show`	(default value). Background and borders shown on empty cells

Default Value: `show`
JavaScript syntax: e.g. *object*`.style.emptyCells="hide"`
Inherited: Yes

Animatable: No

filter

[CSSPropertyFilter]

The CSS (CSS3) `filter` property applies visual effects like grayscale, blur and saturation to an element (usually an <img> element).

Valid property values (other than inherit and initial) are:

Value	Description
`blur` (*length*)	Blur added. Length needs to be in px, e.g. `5px`. Larger value creates more blur. If no value is specified then `0` is used which leaves image unaltered
`brightness` (*%* or *number*)	Brightness adjusted. `0%` (or `0`) makes completely black, `100%` (or `1`) leaves image unaltered and is the default. Values over `100%` will add to brightness.
`contrast` (*%* or *number*)	Contrast adjusted. `0%` (or `0`) makes completely black, `100%` (or `1`) leaves image unaltered and is the default. Values over `100%` will add to contrast.
`drop-shadow` (*h-shadow v-shadow blur spread color*)	Drop shadow applied: *h-shadow* (required). Value (in px) for horizontal shadow (negative values have shadow on left of image *v-shadow* (required). Value (in px) for horizontal shadow (negative values have shadow above image *blur* (optional). Value (in px) adding blur effect to shadow, see above *spread* (optional). Value (in px) which causes shadow to expand (positive) or shrink (negative). Some browsers do not support this parameter *color* (optional). Adds CSS colour to the shadow, default depends on browser but is usually black.
`grayscale` (*%* or *number*)	Converts to grayscale. `0%` (or `0`) is default and leaves image unaltered, `100%` (or `1`) makes image completely gray (i.e. black and white).
`hue-rotate` (*angle*)	Applies a hue rotation to image (see CSS colours for more details of hues). Angle needs to be in degrees, e.g. `20deg`. Default is `0deg`, which leaves image unaltered
`invert` (*%* or *number*)	Inverts colours/brightness in image. `0%` (or `0`) is default and leaves image unaltered, `100%` (or `1`) completely inverts colouring.
`none`	(default value). No effect applied
`opacity` (% or *number*)	Sets opacity level (i.e. degree of transparency) for image. `0%` (or `0`) is completely transparent. `100%` (or `1`) is default and leaves image unaltered (no transparency). Is similar to opacity property.
`saturate` (% or *number*)	Saturates image. `0%` (or `0`) makes image completely unsaturated. 100% (or 1) is default and leaves image unaltered. Values over `100%` (or `1`) add saturation.
`sepia` (% or *number*)	Converts image to sepia. `0%` (or `0`) is default and leaves image unaltered, 100% (or 1) makes image completely sepia (i.e. old photography style black and white).

`url(`*filename*`)`	Takes the location of an XML file that specifies an SVG filter (including potentially an anchor to a specific filter element, e.g.: `filter: url(`*svg-url#svg-element-id*`)`

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.filter="grayscale(90%)"`
Inherited: No
Animatable: Yes

flex

[CSSPropertyFlex]

The CSS (CSS3) `flex` property is a shorthand property for setting certain flex properties for an element.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- flex-grow
- flex-shrink
- flex-basis

Default Value: `0 1 auto`
JavaScript syntax: e.g. *object*`.style.flex="1"`
Inherited: No
Animatable: *See individual properties*

flex-basis

[CSSPropertyFlexBasis]

The CSS (CSS3) `flex-basis` property indicates the initial length of a flexible element. If an element is not a flexible element then this property has no effect.

Valid property values (other than inherit and initial) are:

Value	Description
number	A length unit or percentage specifying initial length of flexible item
`auto`	(default value). Length is equal to length of flexible item or if not specified is set according to context

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.flexBasis="100px"`
Inherited: No
Animatable: Yes

flex-direction

[CSSPropertyFlexDirection]

The CSS (CSS3) `flex-direction` property indicates the direction of a flexible element. If an element is not a flexible element then this property has no effect.

Valid property values (other than inherit and initial) are:

Value	Description
`column`	Items are displayed vertically, as a column
`column-reverse`	Items are displayed vertically, but in reverse order
`row`	(default value). Items are displayed horizontally, as a row
`row-reverse`	Items are displayed horizontally, but in reverse order

Default Value: `row`
JavaScript syntax: e.g. *object*`.style.flexDirection="column"`
Inherited: No
Animatable**:** No

flex-flow

[CSSPropertyFlexFlow]

The CSS (CSS3) `flex-flow` property is a shorthand property for setting certain flex properties for a flex element.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- flex-direction
- flex-wrap

Default Value: `row nowrap`
JavaScript syntax: e.g. *object*`.style.flexFlow="column nowrap"`
Inherited: No
Animatable**:** *See individual properties*

flex-grow

[CSSPropertyFlexGrow]

The CSS (CSS3) `flex-grow` property indicates how much an element will grow relative to the rest of the flexible items in a container. If an element is not a flexible element then this property has no effect.

Valid property values (other than inherit and initial) are:

Value	Description
number	Specifies how much an item will grow relative to rest of flexible items

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.flexGrow="4"`

Inherited: No
Animatable: Yes

flex-shrink

[CSSPropertyFlexShrink]

The CSS (CSS3) `flex-shrink` property indicates how much an element will shrink relative to the rest of the flexible items in a container. If an element is not a flexible element then this property has no effect. Note: for some browsers, the resulting sizes of elements can be less intuitive than using the apparently analogous fractional value for the flex-grow property.

Valid property values (other than inherit and initial) are:

Value	Description
number	Specifies how much an item will shrink relative to rest of flexible items

Default Value: `1`
JavaScript syntax: e.g. *object*`.style.flexShrink="4"`
Inherited: No
Animatable: Yes

flex-wrap

[CSSPropertyFlexWrap]

The CSS (CSS3) `flex-wrap` property indicates whether flexible items should wrap. If an element is not a flexible element then this property has no effect.

Valid property values (other than inherit and initial) are:

Value	Description
`nowrap`	(default value). Will not wrap
`wrap`	Will wrap if necessary
`wrap-reverse`	Will wrap if necessary, but in reverse order

Default Value: `nowrap`
JavaScript syntax: e.g. *object*`.style.flexWrap="wrap-reverse"`
Inherited: No
Animatable: No

float

[CSSPropertyFloat]

The CSS (CSS1) `float` property indicates whether an element should float. If the element is absolutely positioned then the float property is ignored.

Valid property values (other than inherit and initial) are:

Value	Description
`left`	Element floats to left
`none`	(default value). Element not floated (i.e. displays exactly where it occurs in text)
`right`	Element floats to right

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.cssFloat="right"`
Inherited: No
Animatable: No

font

[CSSPropertyFont]

The CSS (CSS1) `font` property is a shorthand property for setting the font properties of an element.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- font-style
- font-variant
- font-weight
- font-size i.e. line height (required)
- font-family (required)

Default Value: *See individual properties*
JavaScript syntax: e.g. *object*`.style.font="italic 12px arial"`
Inherited: Yes
Animatable: *See individual properties*

font-family

[CSSPropertyFontFamily]

The CSS (CSS1) `font-family` property indicates the font to be used for an element. It can include several values, separated by commas (typically starting from more specific fonts and ending with more generic fonts). If the browser doesn't support the first font in the list, it tries the next one etc. Font family names can be:

- Specific, e.g. `"Times New Roman"`, `Arial`; or
- Generic, e.g. `serif`, `cursive`

Valid property values (other than inherit and initial) are:

Value	Description
family-name / *generic-family* / *list*	A prioritised list of specific or generic font family names. Note: if a font name contains a space then it must be enclosed in quotes

Default Value:	*Depends on browser*
JavaScript syntax:	e.g. *object*`.style.fontFamily="Verdana,serif"`
Inherited:	Yes
Animatable**:**	No

font-size

[CSSPropertyFontSize]

The CSS (CSS1) `font-size` property indicates the size of the font to be used for an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Fixed size as a CSS length (px, cm, …)
%	Set as a percentage of parent element's font size
`large`	Large size
`larger`	Larger size than parent element's font size
`medium`	(default value). Medium size
`small`	Small size
`smaller`	Smaller size than parent element's font size
`x-large`	X-large size
`xx-large`	XX-large size
`x-small`	X-small size
`xx-small`	XX-small size

Default Value:	`medium`
JavaScript syntax:	e.g. *object*`.style.fontSize="12px"`
Inherited:	Yes
Animatable**:**	Yes

font-size-adjust

[CSSPropertyFontSizeAdjust]

The CSS (CSS3) `font-size-adjust` property gives better control of the font size than is provided by the font-size property alone, when the first font selected in the font-family property is not available. All fonts have an "aspect value" which is the size-difference between the lowercase "x" and the uppercase "X". If the browser is told this value then it can figure out what font-size to use when displaying text from the entry in the font-family property that is actually used.

Valid property values (other than inherit and initial) are:

Value	Description
number	A value indicating the aspect value to use
`none`	(default value). No adjustment applied to font size

Default Value:	`none`
JavaScript syntax:	e.g. *object*`.style.fontSizeAdjust="0.58"`
Inherited:	Yes

Animatable: Yes

font-stretch

[CSSPropertyFontStretch]

The CSS (CSS3) `font-stretch` property makes text in an element narrower or more stretched out than usual.

Valid property values (other than inherit and initial) are:

Value	Description
`condensed`	Narrower than semi-condensed
`expanded`	Wider than semi-expanded
`extra-condensed`	Narrower than condensed
`extra-expanded`	Wider than expanded
`normal`	(default value). No adjustment applied
`semi-condensed`	Narrower than normal
`semi-expanded`	Wider than normal
`ultra-condensed`	Narrower than extra-condensed
`ultra-expanded`	Wider than extra-expanded

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.fontStretch="condensed"`
Inherited: Yes
Animatable: Yes

font-style

[CSSPropertyFontStyle]

The CSS (CSS1) `font-style` property indicates the font style to use for text in an element.

Valid property values (other than inherit and initial) are:

Value	Description
`italic`	Italic text
`normal`	(default value). Normal text
`oblique`	Oblique text

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.fontStyle="italic"`
Inherited: Yes
Animatable: No

font-variant

[CSSPropertyFontVariant]

The CSS (CSS1) `font-variant` property indicates whether text should be in a small-caps font (in which all lowercase letters are converted to slightly smaller uppercase letters).

Valid property values (other than inherit and initial) are:

Value	Description
`normal`	(default value). Normal text
`small-caps`	Text is shown in a small-caps font

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.fontVariant="small-caps"`
Inherited: Yes
Animatable: No

font-weight

[CSSPropertyFontWeight]

The CSS (CSS1) `font-weight` property indicates how thick or thin (i.e. bold or not) characters should be in the text of an element.

Valid property values (other than inherit and initial) are:

Value	Description
`100, 200, 300, 400, 500, 600, 700, 800, 900`	Number selects from very thin to very thick (400 is normal, 700 is bold)
`bold`	Bold text
`bolder`	Very bold text
`lighter`	Lighter text
`normal`	(default value). Normal text

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.fontWeight="bold"`
Inherited: Yes
Animatable: Yes

hanging-punctuation

[CSSPropertyHangingPunctuation]

The CSS (CSS3) `hanging-punctuation` property indicates whether a punctuation mark can be placed outside the box at the start or end of a full line of text. At the time of writing many major browsers did not support this property.

Valid property values (other than inherit and initial) are:

Value	Description
`allow-end`	Punctuation may hang outside end edge of all lines
`first`	May hang outside start edge
`force-end`	May hang outside end edge of all lines (and will be forced to do so if

	justification applies to the line)
`last`	May hang outside end edge
`none`	(default value). Punctuation mark cannot be placed outside line box

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.hangingPunctuation="last"`
Inherited: Yes
Animatable: No

height

[CSSPropertyHeight]

The CSS (CSS1) `height` property indicates the height of an element (excluding padding, borders and margins). It is overridden by the min-height or max-height properties, if either of them are present.

Valid property values (other than inherit and initial) are:

Value	Description
length	Height as a CSS length (px, cm, …)
%	Height of element defined as a percentage of height of its containing block
`auto`	(default value). Browser determines height

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.height="200px"`
Inherited: No
Animatable: Yes

justify-content

[CSSPropertyJustifyContent]

The CSS (CSS3) `justify-content` property indicates how to align a flexible container's items when the items do not use all available space along the horizontal axis.

Valid property values (other than inherit and initial) are:

Value	Description
`center`	Items are centred
`flex-start`	(default value). Items positioned at start of container
`flex-end`	Items positioned at end of container
`space-around`	Items positioned with space before, between and after lines
`space-between`	Items positioned with space between lines

Default Value: `flex-start`
JavaScript syntax: e.g. *object*`.style.justifyContent="space-around"`
Inherited: No
Animatable: Yes

left

[CSSPropertyLeft]

The CSS (CSS2) `left` property sets, for absolutely positioned elements, the left edge of an element relative to the corresponding edge of its nearest positioned ancestor. If such an element has no positioned ancestors then it uses the document body and moves along with the page scrolling. A 'positioned' element is one whose position is anything other than static.

Valid property values (other than inherit and initial) are:

Value	Description
length	Sets edge position as a CSS length. Can be negative
%	Sets edge position in % of containing element. Can be negative
`auto`	(default value). Browser calculates edge position

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.left="10px"`
Inherited: No
Animatable: Yes

letter-spacing

[CSSPropertyLetterSpacing]

The CSS (CSS1) `letter-spacing` property identifies the amount of space between consecutive text characters.

Valid property values (other than inherit and initial) are:

Value	Description
length	Amount of extra space as a CSS length between characters. Can be negative
`normal`	(default value). No extra space between characters

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.letterSpacing="4px"`
Inherited: Yes
Animatable: Yes

line-height

[CSSPropertyLineHeight]

The CSS (CSS1) `line-height` property identifies the height of lines of text.

Valid property values (other than inherit and initial) are:

Value	Description

length	Line height is a fixed height defined by this CSS length
number	Line height set as a multiple of the current font size
%	Line height set as a percentage of the current font size
`normal`	(default value). Line height is normal (given the relevant font size)

Default Value: normal
JavaScript syntax: e.g. *object*`.style.lineHeight="15px"`
Inherited: Yes
Animatable**:** Yes

list-style

[CSSPropertyListStyle]

The CSS (CSS1) `list-style` property is a shorthand property combining up to 3 list properties.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear) are:

- list-style-type
- list-style-position
- list-style-image

Default Value: `disc outside none`
JavaScript syntax: e.g. *object*.style`.listStyle="decimal inside"`
Inherited: Yes
Animatable**:** See *individual properties*

list-style-image

[CSSPropertyListStyleImage]

The CSS (CSS1) `list-style-image` property identifies an image that should be used as the list-item marker.

Valid property values (other than inherit and initial) are:

Value	Description
`none`	(default value). No image used. Instead, the type of list marker used is defined by the list-style-type property
url	URL path defining the image to be used as the list-item marker

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.listStyleImage="url('picture.gif')"`
Inherited: Yes
Animatable**:** No

list-style-position

[CSSPropertyListStylePosition]

The CSS (CSS1) `list-style-position` property identifies whether a list marker (e.g. a bullet character or (a), (b), (c) etc.) is inside or outside the relevant content container.

Valid property values (other than inherit and initial) are:

Value	Description
`inside`	Marker and text are indented and appear inside the relevant content container
`outside`	(default value). Marker is kept outside the relevant content container

Default Value: `outside`
JavaScript syntax: e.g. *object*`.style.listStylePosition="inside"`
Inherited: Yes
Animatable**:** No

list-style-type

[CSSPropertyListStyleType]

The CSS (CSS1) `list-style-type` property identifies the type of list-item marker used for a specified list.

Valid property values (other than inherit and initial) are:

Value	Description
`armenian`	Armenian numbering
`circle`	A circle
`cjk-ideographic`	Plain ideographic numbers
`decimal`	A number
`decimal-leading-zero`	A number with leading zeros (i.e. 01, 02, … if reaches beyond 9 but not beyond 99
`disc`	(default value). A filled circle
`georgian`	Georgian numbering
`hebrew`	Hebrew numbering
`hiragana`	Hiragana numbering
`hiragana-iroha`	Hiragana iroha numbering
`katakana`	Katakana numbering
`katakana-iroha`	Katakana iroha numbering
`lower-alpha`	I.e. a, b, c, …
`lower-greek`	I.e. lower case greek, α, β, γ, …
`lower-latin`	I.e. a, b, c, …
`lower-roman`	I.e. i, ii, iii, …
`none`	No marker
`square`	Square marker
`upper-alpha`	I.e. A, B, C, …
`upper-latin`	I.e. A, B, C, …
`upper-roman`	I.e. I, II, III, …

Default Value: `disc`
JavaScript syntax: e.g. *object*`.style.listStyleType="decimal"`
Inherited: Yes
Animatable**:** No

margin

[CSSPropertyMargin]

The CSS (CSS1) `margin` property is a shorthand property combining all four margin properties. The individual margin widths can be set separately using margin-bottom, margin-left, margin-right and margin-top. As with some other aggregate edge properties, up to four parameter values can be supplied (and if more than one is supplied then the properties are applied to individual borders as described here).

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements included in the *margin* property are:

Value	Description
length	Any specified margin size as a CSS length
%	Margin specified as percentage of width of container
`auto`	Browser calculates margin

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.margin="25px 30px"`
Inherited: No
Animatable**:** See *individual properties*

margin-bottom

[CSSPropertyMarginBottom]

The CSS (CSS1) `margin-bottom` property sets the width of the bottom margin of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified margin size as a CSS length
%	Margin specified as percentage of width of container
`auto`	Browser calculates margin

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.marginBottom="30px"`
Inherited: No
Animatable**:** Yes

margin-left

[CSSPropertyMarginLeft]

The CSS (CSS1) `margin-left` property sets the width of the left margin of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified margin size as a CSS length
%	Margin specified as percentage of width of container
`auto`	Browser calculates margin

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.marginLeft="30px"`
Inherited: No
Animatable**:** Yes

margin-right

[CSSPropertyMarginRight]

The CSS (CSS1) `margin-right` property sets the width of the right margin of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified margin size as a CSS length
%	Margin specified as percentage of width of container
`auto`	Browser calculates margin

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.marginRight="30px"`
Inherited: No
Animatable**:** Yes

margin-top

[CSSPropertyMarginTop]

The CSS (CSS1) `margin-top` property sets the width of the top margin of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified margin size as a CSS length
%	Margin specified as percentage of width of container
`auto`	Browser calculates margin

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.marginTop="30px"`
Inherited: No
Animatable**:** Yes

max-height

[CSSPropertyMaxHeight]

The CSS (CSS2) `max-height` property sets the maximum height an element can become. It overrides the height property.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length
%	Defined as a percentage of that of the containing block
`none`	(default value). No limit

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.maxHeight="200px"`
Inherited: No
Animatable**:** Yes

max-width

[CSSPropertyMaxWidth]

The CSS (CSS2) `max-width` property sets the maximum width an element can become. It overrides the width property.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length
%	Defined as a percentage of that of the containing block
`none`	(default value). No limit

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.maxWidth="300px"`
Inherited: No
Animatable**:** Yes

min-height

[CSSPropertyMinHeight]

The CSS (CSS2) `min-height` property sets the minimum height an element can become. It overrides the height property and the max-height property.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length
%	Defined as a percentage of that of the containing block

`none`	(default value). No limit

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.minHeight="100px"`
Inherited: No
Animatable: Yes

min-width

[CSSPropertyMinWidth]

The CSS (CSS2) `min-width` property sets the minimum width an element can become. It overrides the width property and the max-width property.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length
%	Defined as a percentage of that of the containing block
`none`	(default value). No limit

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.minWidth="150px"`
Inherited: No
Animatable: Yes

nav-down, nav-index, nav-left, nav-right, nav-up

[CSSPropertyNav]

At the time of writing the CSS (CSS3) `nav-down`, `nav-index`, `nav-left`, `nav-right` and `nav-up` properties appear to be supported by very few browsers, so are not explained further here. They indicate where to navigate to when using the down arrow, left arrow, right arrow and up arrow keys respectively.

The `nav-index` property specifies the sequential navigation order (i.e. the 'tabbing order') for an element.

opacity

[CSSPropertyOpacity]

The CSS (CSS3) `opacity` property sets the degree of opacity (transparency) of an element. 0 is completely transparent, 1 is completely non-transparent. Note also sets the transparency of all relevant child elements (if you don't want this to happen then use RGBA colouring).

Valid property values (other than inherit and initial) are:

Value	Description
number	From `0.0` to `1.0`

Default Value:	`1`
JavaScript syntax:	e.g. *object*`.style.opacity="0.6"`
Inherited:	No
Animatable**:**	Yes

order

[CSSPropertyOrder]

The CSS (CSS3) `order` property indicates the order of a flexible item relative to other flexible items inside the same container.

Valid property values (other than inherit and initial) are:

Value	Description
number	Integer

Default Value:	`0`
JavaScript syntax:	e.g. *object*`.style.order="2"`
Inherited:	No
Animatable**:**	Yes

orphans

[CSSPropertyOrphans]

The CSS (CSS3) `orphans` property indicates the minimum number of lines of a paragraph that can be left on an old page. It works primarily with paged media, in which content is split into pages.

Valid property values (other than inherit and initial) are:

Value	Description
number	Integer

Default Value:	`2`
JavaScript syntax:	e.g. *object*`.style.orphans="3"`
Inherited:	No
Animatable**:**	Yes

outline

[CSSPropertyOutline]

The CSS (CSS2) `outline` property is a shorthand property combining (up to) 3 of the outline properties.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- outline-color
- outline-style
- outline-width

Default Value: `invert none medium`
JavaScript syntax: e.g. *object*`.style.outline="2px solid red"`
Inherited: No
Animatable**:** See *individual properties*

outline-color

[CSSPropertyOutlineColor]

The CSS (CSS2) `outline-color` property sets the color of the outline of an element (i.e. a line that is drawn around the element, outside its borders, usually to make the element stand out relative to other elements.

Valid property values (other than inherit and initial) are:

Value	Description
color	A CSS colour
`invert`	(default value). Inverts colour

Default Value: `invert`
JavaScript syntax: e.g. *object*`.style.outlineColor="red"`
Inherited: No
Animatable**:** Yes

outline-offset

[CSSPropertyOutlineOffset]

The CSS (CSS2) `outline-offset` property sets the amount of space between an element's outline and the edge or border of the element.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.outlineOffset="10px"`
Inherited: No
Animatable**:** Yes

outline-style

[CSSPropertyOutlineStyle]

The CSS (CSS2) `outline-style` property specifies the style to be used for the outline of an element.

Valid property values (other than inherit and initial) are:

Value	Description
`dashed`	Dashed outline
`dotted`	Dotted outline
`double`	Double outline
`groove`	Effect depends on outline-color value
`hidden`	Hidden outline
`inset`	Effect depends on outline-color value
`none`	(default value). No outline
`outset`	Effect depends on outline-color value
`ridge`	Effect depends on outline-color value
`solid`	Solid outline

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.outlineStyle="ridge"`
Inherited: No
Animatable**:** Yes

outline-width

[CSSPropertyOutlineWidth]

The CSS (CSS2) `outline-width` property sets the width of an element's outline (outside the edge or border of the element). Note: an element needs to have an outline (so you need to set the outline-style property to something other than none) before the width of the outline can be set.

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.outlineWidth="3px"`
Inherited: No
Animatable**:** Yes

overflow

[CSSPropertyOverflow]

The CSS (CSS2) `overflow` property indicates what happens when content overflows an element's box.

Valid property values (other than inherit and initial) are:

Value	Description

auto	Overflow is clipped and a scroll-bar should typically be added
hidden	Overflow is clipped. No scroll-bar is added, so rest of content is effectively invisible
scroll	Overflow is clipped and a scroll-bar is added
visible	(default value). Overflow is not clipped and is rendered outside the element's box

Default Value: visible
JavaScript syntax: e.g. *object*.style.overflow="scroll"
Inherited: No
Animatable: No

overflow-x

[CSSPropertyOverflowX]

The CSS (CSS2) overflow-x property indicates what to with left/right edges of content overflowing an element's box.

Valid property values (other than inherit and initial) are:

Value	Description
auto	Overflow is clipped and a scroll-bar should typically be added
hidden	Overflow is clipped. No scroll-bar is added, so rest of content is effectively invisible
scroll	Overflow is clipped and a scroll-bar is added
visible	(default value). Overflow is not clipped and may be rendered outside the element's box

Default Value: visible
JavaScript syntax: e.g. *object*.style.overflowX="scroll"
Inherited: No
Animatable: No

overflow-y

[CSSPropertyOverflowY]

The CSS (CSS2) overflow-y property indicates what to with top/bottom edges of content overflowing an element's box.

Valid property values (other than inherit and initial) are:

Value	Description
auto	Overflow is clipped and a scroll-bar should typically be added
hidden	Overflow is clipped. No scroll-bar is added, so rest of content is effectively invisible
scroll	Overflow is clipped and a scroll-bar is added
visible	(default value). Overflow is not clipped and may be rendered outside the element's box

Default Value: `visible`
JavaScript syntax: e.g. *object*`.style.overflowY="scroll"`
Inherited: No
Animatable: No

padding

[CSSPropertyPadding]

The CSS (CSS1) `padding` property is a shorthand property combining the 4 padding sub-properties. The individual padding widths can be set separately using padding-bottom, padding-left, padding-right and padding-top. As with some other aggregate edge properties, up to four parameter values can be supplied (and if more than one is supplied then the properties are applied to individual borders as described here).

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements included in the *margin* property are:

Value	Description
length	Any specified padding size as a CSS length
%	Padding size specified as percentage of width of container

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.padding="20px 30px"`
Inherited: No
Animatable: See *individual properties*

padding-bottom

[CSSPropertyPaddingBottom]

The CSS (CSS1) `padding-bottom` property sets the width of the bottom padding (space) of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
%	As percentage of width of containing element

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.paddingBottom="4px"`
Inherited: No
Animatable: Yes

padding-left

[CSSPropertyPaddingLeft]

The CSS (CSS1) `padding-left` property sets the width of the left padding (space) of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
%	As percentage of width of containing element

Default Value: 0
JavaScript syntax: e.g. *object*`.style.paddingLeft="4px"`
Inherited: No
Animatable**:** Yes

padding-right

[CSSPropertyPaddingRight]

The CSS (CSS1) `padding-right` property sets the width of the right padding (space) of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
%	As percentage of width of containing element

Default Value: 0
JavaScript syntax: e.g. *object*`.style.paddingRight="4px"`
Inherited: No
Animatable**:** Yes

padding-top

[CSSPropertyPaddingTop]

The CSS (CSS1) `padding-top` property sets the width of the top padding (space) of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Any specified thickness as a CSS length
%	As percentage of width of containing element

Default Value: 0
JavaScript syntax: e.g. *object*`.style.paddingTop="4px"`
Inherited: No
Animatable**:** Yes

page-break-after

[CSSPropertyPageBreakAfter]

The CSS (CSS2) `page-break-after` property specifies whether a page break should occur after an element. It cannot be used on an empty element or on absolutely positioned elements.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Page breaking defined automatically
`always`	Page break always inserted after element
`avoid`	Where possible avoids a page break after the element
`left`	Page break is inserted after element in a manner that results in the next page being formatted as a left page
`right`	Page break is inserted after element in a manner that results in the next page being formatted as a right page

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.pageBreakAfter="always"`
Inherited: No
Animatable: No

page-break-before

[CSSPropertyPageBreakBefore]

The CSS (CSS2) `page-break-before` property specifies whether a page break should occur before an element. It cannot be used on an empty element or on absolutely positioned elements.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Page breaking defined automatically
`always`	Page break always inserted before element
`avoid`	Where possible avoids a page break before the element
`left`	Page break is inserted before element in a manner that results in the next page being formatted as a left page
`right`	Page break is inserted before element in a manner that results in the next page being formatted as a right page

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.pageBreakBefore="always"`
Inherited: No
Animatable: No

page-break-inside

[CSSPropertyPageBreakInside]

The CSS (CSS2) `page-break-inside` property specifies whether a page break is allowed inside an element. It cannot be used on absolutely positioned elements.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Page breaking defined automatically
`avoid`	Where possible avoids a page break inside the element

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.pageBreakInside="avoid"`
Inherited: No
Animatable: No

perspective

[CSSPropertyPerspective]

The CSS (CSS3) `perspective` property indicates how far a 3D element is notionally placed behind the screen. The property applies to the child elements not the original element itself to which this property is attached.

Valid property values (other than inherit and initial) are:

Value	Description
`length`	Page breaking defined automatically
`none`	(default value). Same as 0, i.e. no perspective set

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.perspective="30px"`
Inherited: No
Animatable: Yes

perspective-origin

[CSSPropertyPerspectiveOrigin]

The CSS (CSS3) `perspective-origin` property indicates where a 3D element is notionally placed, based on the *x*-axis and *y*-axis. The property applies to the child elements not the original element itself to which this property is attached. It needs to be used in conjunction with the perspective property and only affects 3D transformed elements.

Valid property values (other than inherit and initial) are:

Value	Description
x-axis y-axis	*x-axis* defines where the view is placed on the *x-axis*, *y-axis* where it is placed on the *y-axis*. Possible values for *x-axis* are: - `left`

	- `center` - `right` - *length* (a CSS length) - *%* (of element size) Possible values for *y-axis* are: - `top` - `center` - `bottom` - *length* (a CSS length) - *%* (of element size)

Default Value: `50% 50%`
JavaScript syntax: e.g. *object*`.style.perspectiveOrigin="20px 40px"`
Inherited: No
Animatable**:** Yes

position

[CSSPropertyPosition]

The CSS (CSS2) `position` property indicates how an element should be positioned.

Valid property values (other than inherit and initial) are:

Value	Description
`absolute`	The element is positioned relative to its first positioned (i.e. not static) ancestor
`fixed`	The element is positioned in a fixed position relative to the browser window
`relative`	The element is positioned relative to its normal position
`static`	(default value). The element is positioned (relative to others) in the order in which it appears in the document flow

Default Value: `static`
JavaScript syntax: e.g. *object*`.style.position="fixed"`
Inherited: No
Animatable**:** No

quotes

[CSSPropertyQuotes]

The CSS (CSS2) `quotes` property indicates how quotation marks should be rendered in the text of an element.

Valid property values (other than inherit and initial) are:

Value	Description
`none`	Open-quote and close-quote elements within an element (i.e. the

	<q> and </q> of a <q> element) will not produce any quotation marks
string1 string2 string3 string4	*String1* and *string2* define the first level of quotation embedding (opening and closing respectively), *string3* and *string4* (if present) the next level of embedding etc.

Default Value: N/A
JavaScript syntax: e.g. *object*`.style.quotes="'\u00AB' '\u00BB'"`
Inherited: No
Animatable**:** No

Common quotation mark characters include:

Representation	Character code used in Javascript syntax	Name
"	`\u0022`	Double quotation mark
'	`\u0027`	Single quotation mark
‘	`\u2018`	Left single high quotation mark
’	`\u2019`	Right single high quotation mark
“	`\u201C`	Left double high quotation mark
”	`\u201D`	Right double high quotation mark
‹	`\u2039`	Left single angle quotation mark
›	`\u203A`	Right single angle quotation mark
«	`\u00AB`	Left double angle quotation mark
»	`\u00BB`	Right double angle quotation mark
„	`\u201E`	Right double low quotation mark

resize

[CSSPropertyResize]

The CSS (CSS3) `resize` property indicates whether an element can be resized by the user. Some major browsers do not support this property.

Valid property values (other than inherit and initial) are:

Value	**Description**
`both`	User can resize height and width of element
`horizontal`	User can resize width of element
`none`	(default value). User cannot resize element
`vertical`	User can resize height of element

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.resize="horizontal"`
Inherited: No
Animatable**:** No

right

[CSSPropertyRight]

The CSS (CSS2) `right` property sets, for absolutely positioned elements, the right edge of an element relative to the corresponding edge of its nearest positioned ancestor. If such an element has no positioned ancestors then it uses the document body and moves along with page scrolling. A 'positioned' element is one whose position is anything other than static.

Valid property values (other than inherit and initial) are:

Value	Description
length	Sets edge position as a CSS length. Can be negative
%	Sets edge position in % of containing element. Can be negative
`auto`	(default value). Browser calculates edge position

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.right="10px"`
Inherited: No
Animatable: Yes

tab-size

[CSSPropertyTabSize]

The CSS (CSS3) `tab-size` property indicates size (length) of space used for the tab character.

Valid property values (other than inherit and initial) are:

Value	Description
number	Number of space characters displayed for each tab character
length	Length of tab character (not supported by major browsers)
`none`	User cannot resize element
`vertical`	User can resize height of element

Default Value: `8`
JavaScript syntax: e.g. *object*`.style.tabSize="12"`
Inherited: No
Animatable: No

table-layout

[CSSPropertyTableLayout]

The CSS (CSS2) `table-layout` property indicates the algorithm used to define the table layout.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Layout set automatically, with column width set by widest unbreakable content. This can render more slowly as it means all content needs to be read before the layout can be determined)
`fixed`	Layout set only by reference to table's width and width of columns,

	i.e. not by what is in each cell. This can render faster as the browser can begin to display the table as soon as the first row has been received

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.tableLayout="fixed"`
Inherited: No
Animatable: No

text-align

[CSSPropertyTextAlign]

The CSS (CSS1) `text-align` property indicates how text in an element should be aligned.

Valid property values (other than inherit and initial) are:

Value	Description
`center`	Centres the text
`justify`	Justifies text, i.e. stretches lines to encompass whole width. The precise way justification then works is set by the text-justify property.
`left`	Aligns text to left
`right`	Aligns text to right

Default Value: `left` if direction is `ltr` (left-to-right), `right` if direction is `rtl` (right-to-left)
JavaScript syntax: e.g. *object*`.style.textAlign="justify"`
Inherited: Yes
Animatable: No

text-align-last

[CSSPropertyTextAlignLast]

The CSS (CSS3) `text-align-last` property indicates how the last line of text in an element should be aligned.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Last line is aligned left
`center`	Last line of text is centred
`end`	Last line is aligned to end of line (right if direction is left-to-right, left if direction is right-to-left)
`justify`	Justifies last line of text, i.e. stretches lines to encompass whole width (rarely how text is formatted in practice)
`left`	Aligns last line of text to left
`start`	Last line is aligned to start of line (left if direction is left-to-right, right if direction is right-to-left)

`right`	Aligns last line of text to right

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.textAlignLast="right"`
Inherited: Yes
Animatable**:** No

text-decoration

[CSSPropertyTextDecoration]

The CSS (CSS1) `text-decoration` property indicates the 'decoration' (e.g. underlining) added to text. Note in CSS3 the *text-decoration* property is supposed to be a shorthand property covering text-decoration-line, text-decoration-color and text-decoration-style but at the time of writing this interpretation was not supported by major browsers.

Valid property values (other than inherit and initial) are:

Value	Description
`line-through`	Last line is aligned left
`none`	(default value). Normal text, without decoration
`overline`	Add line above text
`underline`	Add line below text

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.textDecoration="line-through"`
Inherited: No
Animatable**:** No

text-decoration-color

[CSSPropertyTextDecorationColor]

The CSS (CSS3) `text-decoration-color` property indicates the colour of the text decoration added to a given piece of text. For this property to have an effect, a visible text-decoration needs to have been applied to the text.

Valid property values (other than inherit and initial) are:

Value	Description
color	A CSS colour

Default Value: *Colour of text*
JavaScript syntax: e.g. *object*`.style.textDecorationColor="red"`
Inherited: No
Animatable**:** Yes

text-decoration-line

[CSSPropertyTextDecorationLine]

The CSS (CSS3) `text-decoration-line` property indicates the type of line of the text decoration added to a given piece of text. For this property to have an effect, a visible text-decoration needs to have been applied to the text. Multiple values can be combined (e.g. underline and overline)

Valid property values (other than inherit and initial) are:

Value	Description
`line-through`	Last line is aligned left
`none`	(default value). Normal text, without decoration
`overline`	Add line above text
`underline`	Add line below text

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.textDecorationLine="line-through"`
Inherited: No
Animatable: No

text-decoration-style

[CSSPropertyTextDecorationStyle]

The CSS (CSS3) `text-decoration-style` property indicates the style of any line in any visible text decoration added to a given piece of text. For this property to have an effect, a visible text-decoration needs to have been applied to the text.

Valid property values (other than inherit and initial) are:

Value	Description
`dashed`	Text decoration displays as a dashed line
`dotted`	Text decoration displays as a dotted line
`double`	Text decoration displays as a double line
`solid`	(default value). Text decoration displays as a single line
`wavy`	Text decoration displays as a wavy line

Default Value: `solid`
JavaScript syntax: e.g. *object*`.style.textDecorationStyle="dotted"`
Inherited: No
Animatable: No

text-indent

[CSSPropertyTextIndent]

The CSS (CSS1) `text-indent` property determines the indentation applied to the first line of text in an element. Negative values are allowed (making it possible in effect to indent all but the first line, if the element is sized appropriately).

Valid property values (other than inherit and initial) are:

Value	Description
length	A CSS length
%	Indentation defined in terms of % of width of the parent element

Default Value: `0`
JavaScript syntax: e.g. *object*`.style.textIndent="20px"`
Inherited: Yes
Animatable**:** Yes

text-justify

[CSSPropertyTextJustify]

The CSS (CSS3) `text-justify` property indicates how text is justified when the text-align property has been set to `justify`.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Browser determines justification
`distribute`	Primarily changes spacing at word separators and at grapheme boundaries in scripts other than connected and cursive scripts
`inter-cluster`	Primarily changes spacing at word separators and at grapheme boundaries in cursive scripts
`inter-ideograph`	Primarily changes spacing at word separators and at grapheme boundaries in connected scripts (i.e. ones that use no word spaces)
`inter-word`	Primarily changes spacing at word separators
`kashida`	Primarily stretches Arabic and related scripts through use of kashida or other calligraphic elongation

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.textJustify="distribute"`
Inherited: Yes
Animatable**:** No

text-overflow

[CSSPropertyTextOverflow]

The CSS (CSS3) `text-overflow` property indicates how text that has overflowed is rendered by the browser.

Valid property values (other than inherit and initial) are:

Value	Description
`clip`	(default value). Text is clipped without further action
`ellipsis`	Clipped text is rendered by an ellipsis, i.e. “…”
string	Clipped text is rendered by the given string

Default Value: `clip`
JavaScript syntax: e.g. *object*`.style.textOverflow="ellipsis"`
Inherited: No
Animatable**:** No

text-shadow

[CSSPropertyTextShadow]

The CSS (CSS3) `text-shadow` property indicates what shadow should be added to text. To add more than one shadow, the property should be set to a comma-separated list of shadows.

Valid property values (other than inherit and initial) are:

Value	Description
`none`	(default value). No shadow
h-shadow v-shadow blur-radius color	2 to 4 parameters: - *h-shadow* (required): position of horizontal shadow (can be negative) - *v-shadow* (required): position of vertical shadow (can be negative) - *blur-radius* (optional): how fuzzy (default is zero) - *color* (optional): CSS colour of shadow (default is colour of text)

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.textShadow="2px 10px 5px blue"`
Inherited: Yes
Animatable**:** Yes

text-transform

[CSSPropertyTextTransform]

The CSS (CSS1) `text-transform` property specifies the capitalisation the browser should use for text.

Valid property values (other than inherit and initial) are:

Value	Description
`capitalize`	First character of each word is capitalised (i.e. transformed to uppercase)
`lowercase`	All characters transformed to lowercase
`none`	(default value). No capitalisation
`uppercase`	All characters transformed to uppercase

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.textTransform="capitalize"`
Inherited: Yes
Animatable**:** No

top

[CSSPropertyTop]

The CSS (CSS2) `top` property sets, for absolutely positioned elements, the top edge of an element relative to the corresponding edge of its nearest positioned ancestor. If such an element has no positioned ancestors then it uses the document body and moves along with the page scrolling. A 'positioned' element is one whose position is anything other than static.

Valid property values (other than inherit and initial) are:

Value	Description
length	Sets edge position as a CSS length. Can be negative
%	Sets edge position in % of containing element. Can be negative
`auto`	(default value). Browser calculates edge position

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.top="10px"`
Inherited: No
Animatable**:** Yes

transform

[CSSPropertyTransform]

The CSS (CSS3) `transform` property applies a 2D or a 3D transformation to an element, e.g. rotate, scale, skew transform or translate (move) an element.

Valid property values (other than inherit and initial) are:

Value	Description
`matrix(`*n1, n2, n3, n4, n5, n6*`)`	2D transformation characterised by 6 values, see below
`matrix3d(`*n1, n2, n3, n4, n5, n6, n7, n8, n9, n10, n11, n12, n13, n14, n15, n16*`)`	3D transformation characterised by 16 values, see below
`none`	(default value). No transformation applied
`perspective(`*n*`)`	A perspective view for a 3D transformed element
`rotate(`*angle*`)`	2D rotation (around origin), *angle* being a CSS angle
`rotate3d(`*x, y, z, angle*`)`	3D rotation (around line through origin), *x*, *y* and *z* being CSS lengths and *angle* being a CSS angle
`rotateX(`*angle*`)`	3D rotation (around *x*-axis), *angle* being a CSS angle
`rotateY(`*angle*`)`	3D rotation (around *y*-axis), *angle* being a CSS angle
`rotateZ(`*angle*`)`	3D rotation (around *z*-axis), *angle* being a CSS angle
`scale(`*x, y*`)`	2D scaling transformation, applied to *x* and *y*-axes, *x* and *y* being numbers
`scale3d(`*x, y, z*`)`	3D scaling transformation, applied to *x*, *y* and *z*-axes, *x*, *y* and *z* being numbers
`scaleX(`*x*`)`	Scaling transformation (stretching / squashing) applied to *x*-axis, *x*

	being numbers
`scaleY(y)`	Scaling transformation (stretching / squashing) applied to *y*-axis, *y* being numbers
`scaleZ(z)`	Scaling transformation (stretching / squashing) applied to *z*-axis, *z* being numbers
`skew(x-angle, y-angle)`	2D skew transformation along *x* and *y*-axes, *x-angle* and *y-angle* being CSS angles
`skewX(angle)`	Skew transformation along *x*-axis, *angle* being a CSS angle
`skewY(angle)`	Skew transformation along *y*-axis, *angle* being a CSS angle
`translate(x, y)`	2D translation, applied to *x* and *y*-axes, *x* and *y* being CSS lengths
`translate3d(x, y, z)`	3D translation, applied to *x*, *y* and *z*-axes, *x*, *y* and *z* being CSS lengths
`translateX(x)`	Translation (movement) applied to *x*-axis, *x* being a CSS length
`translateY(y)`	Translation (movement) applied to *y*-axis, *y* being a CSS length
`translateZ(z)`	Translation (movement) applied to *z*-axis, *z* being a CSS length

Default Value: `none`
JavaScript syntax: e.g. *object*`.style.transform="translateX(30px)"`
Inherited: No
Animatable: Yes

2D transformations

There are 6 types of 2D transformations: `translate()`, `rotate()`, `scale()`, `skewX()`, `skewY()` and `matrix()`. These have the following characteristics:

- `translate()`: moves an element from its current position along the *x* and *y*-axes, but doesn't otherwise change its shape or size
- `rotate()`: rotates an element clockwise (positive) or counter-clockwise (negative), e.g. rotate(20deg)
- `scale()`: increases or decreases the size of an element (parameter is a ratio relative to the original size), first parameter is width (i.e. *x*-axis) scaling, second is height (i.e. *y*-axis) scaling
- `skewX()`: introduces a skew along the *x*-axis, by a given angle (positive or negative)
- `skewY()`: introduces a skew along the *y*-axis, by a given angle (positive or negative)
- `skew(x-angle, y-angle)`: combines skewX(*x-angle*) and skewY(*y-angle*)
- `matrix()`: combines all 2D transform methods into a single method, involving the following parameters: `matrix(`*scale-x*, *skew-y*, *skew-x*, *scale-y*, *translate-x*, *translate-y*`)`

The origin of the transformation is defined by the transform-origin property.

3D transformations

Not currently covered in this page.

transform-origin

[CSSPropertyTransformOrigin]

The CSS (CSS3) `transform-origin` property defines the origin used by the transform property.

Valid property values (other than inherit and initial) are:

Value	Description
x-axis y-axis z-axis	Origin defined by three values, where: *x-axis* can be: - *length* (a CSS length) - *%* (of overall element size) - `left` (origin *x*-coordinate at left end of element) - `centre` (origin *x*-coordinate at centre of element) - `right` (origin *x*-coordinate at right end of element) *y-axis* can be: - *length* (a CSS length) - *%* (of overall element size) - `bottom` (origin *y*-coordinate at bottom end of element) - `centre` (origin *y*-coordinate at centre of element) - `top` (origin *y*-coordinate at right end of element) *z-axis* can be: - *length* (a CSS length)

Default Value: `50% 50% 0`
JavaScript syntax: e.g. *object*`.style.transformOrigin="0 0"`
Inherited: No
Animatable: Yes

transform-style

[CSSPropertyTransformStyle]

The CSS (CSS3) `transform-style` property indicates how nested elements are to be rendered for 3D purposes when using the transform property.

Valid property values (other than inherit and initial) are:

Value	Description
`flat`	(default value). Child elements do not preserve their 3D position
`preserve-3d`	Child elements preserve their own 3D position

Default Value: `flat`
JavaScript syntax: e.g. *object*`.style.transformStyle="preserve-3d"`
Inherited: No
Animatable: No

transition

[CSSPropertyTransition]

The CSS (CSS3) `transition` property is a shorthand property combining the 4 transition sub-properties.

Valid property values (other than inherit and initial) are defined by the elements of the shorthand. Shorthand elements (in the order in which they appear):

- transition-property
- transition-duration
- transition-timing-function
- transition-delay

Default Value: See *individual properties*
JavaScript syntax: e.g. *object*`.style.transition="all 5s"`
Inherited: No
Animatable**:** No

transition-delay

[CSSPropertyTransitionDelay]

The CSS (CSS3) `transition-delay` property indicates when a transition will start.

Valid property values (other than inherit and initial) are:

Value	Description
time	The CSS time to wait before transition effect starts

Default Value: `0s`
JavaScript syntax: e.g. *object*`.style.transitionDelay="5s"`
Inherited: No
Animatable**:** No

transition-duration

[CSSPropertyTransitionDuration]

The CSS (CSS3) `transition-duration` property indicates how long a transition will take to complete.

Valid property values (other than inherit and initial) are:

Value	Description
time	The CSS time a transition will take to complete once started

Default Value: `0s`
JavaScript syntax: e.g. *object*`.style.transitionDuration="4s"`
Inherited: No
Animatable**:** No

transition-property

[CSSPropertyTransitionProperty]

The CSS (CSS3) `transition-property` property identifies the properties that change as part of a transition effect. You should always specify the transition-duration property as well, as otherwise it defaults to zero.

Valid property values (other than inherit and initial) are:

Value	Description
property	Comma separated list of CSS properties to which transition is applied
`all`	(default value). Transition effect is applied to all CSS properties
`none`	Transition effect is applied to no properties

Default Value: `all`
JavaScript syntax: e.g. *object*`.style.transitionProperty="width"`
Inherited: No
Animatable**:** No

transition-timing-function

[CSSPropertyTransitionTimingFunction]

The CSS (CSS3) `transition-timing-function` property identifies the speed curve used for a transition effect.

Valid property values (other than inherit and initial) are:

Value	Description
`cubic-bezier(`*x1*, *x2*, *x3*, *x4*`)`	As defined by cubic Bezier function with parameters *x1*, *x2*, *x3*, *x4* (possible values of each are numerical values from `0` to `1`)
`ease`	(default value). Slow start, then fast, before ending slowly, equivalent to cubic-bezier(0.25,0.1,0.25,1)
`ease-in`	Slow start, equivalent to cubic-bezier(0.42,0,1,1)
`ease-out`	Slow end, equivalent to cubic-bezier(0,0,0.58,1)
`ease-in-out`	Slow start and end, equivalent to cubic-bezier(0.42,0,0.58,1)
`step-start`	Equivalent to `steps(1, start)`
`step-end`	Equivalent to `steps(1, end)`
`steps(int,start|end)`	A stepping function with two parameters. The first parameter specifies the number of intervals in the function (and must be a positive integer, i.e. greater than zero). The second (optional) parameter specifies when the change of values occurs and is either `start` or `end` (if omitted is given the value `end`)

Default Value: `ease`
JavaScript syntax: e.g. *object*`.style.transitionTimingFunction="linear"`
Inherited: No
Animatable**:** No

unicode-bidi

[CSSPropertyUnicodeBidi]

The CSS (CSS2) `unicode-bidi` property indicates whether text direction should be overridden to support multiple languages in the same document. It is used in conjunction with the direction property.

Valid property values (other than inherit and initial) are:

Value	Description
`bidi-override`	Creates an additional level of embedding and reorders depending on direction property
`embed`	Creates an additional level of embedding
`normal`	(default value). No additional level of embedding

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.unicodeBidi="bidi-override"`
Inherited: Yes
Animatable**:** No

user-select

[CSSPropertyUserSelect]

The CSS (CSS3) `user-select` property indicates whether text of an element can be selected. If you (double) click on some text it will typically be selected, and this property stops this happening.

Valid property values (other than inherit and initial) are:

Value	Description
`auto`	(default value). Text can be selected by user (if allowed by browser)
`none`	Text can't be selected by user
`text`	Text can be selected by user

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.userSelect="none"`
Inherited: No
Animatable**:** No

vertical-align

[CSSPropertyVerticalAlign]

The CSS (CSS1) `vertical-align` property indicates the vertical alignment of an element.

Valid property values (other than inherit and initial) are:

Value	Description
length	Raises (positive) or lowers (negative) element by a CSS length
%	Raises of lowers element by percentage of line-height property
`baseline`	(default value). Baseline of element aligned with baseline of parent
`bottom`	Bottom of element aligned with bottom of lowest element on line

middle	Element is placed vertically in middle of parent
sub	Aligns element as if it was a subscript
super	Aligns element as if it was a superscript
text-bottom	Bottom of element aligned with bottom of parent text
text-top	Top of element aligned with top of parent text
top	Top of element aligned with top of highest element on line

Default Value: baseline
JavaScript syntax: e.g. *object*.style.verticalAlign="sub"
Inherited: No
Animatable: Yes

visibility

[CSSPropertyVisibility]

The CSS (CSS2) visibility property indicates whether an element is visible or not. Note: Invisible (hidden) elements still take up some space on a page; if you want to avoid this then set the display property to none.

Valid property values (other than inherit and initial) are:

Value	Description
collapse	Only applies to table elements. Row or column is removed, but table layout is otherwise left unaltered. For non-table elements is equivalent to "hidden"
hidden	Element is hidden (but still takes up space)
visible	(default value). Element is visible

Default Value: visible
JavaScript syntax: e.g. *object*.style.visibility="hidden"
Inherited: No
Animatable: Yes

white-space

[CSSPropertyWhiteSpace]

The CSS (CSS1) white-space property indicates how white-space inside an element should be handled.

Valid property values (other than inherit and initial) are:

Value	Description
normal	(default value). Sequences of whitespace collapsed into a single whitespace and text will wrap when necessary
nowrap	Sequences of whitespace collapsed into a single whitespace but text will not wrap until a tag occurs
pre	Whitespace is preserved by browser and text will only wrap on line breaks (i.e. akin to <pre> tag in HTML)

`pre-line`	Sequences of whitespace collapsed into a single whitespace and text will wrap when necessary and on line breaks
`pre-wrap`	Whitespace is preserved by browser and text will wrap when necessary and on line breaks

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.whiteSpace="hidden"`
Inherited: Yes
Animatable**:** No

widows

[CSSPropertyWidows]

The CSS (CSS3) `widows` property indicates the minimum number of lines of a paragraph that can fall to a new page. It works primarily with paged media, in which content is split into pages.

Valid property values (other than inherit and initial) are:

Value	Description
number	Integer

Default Value: `2`
JavaScript syntax: e.g. *object*`.style.widows="3"`
Inherited: No
Animatable**:** Yes

width

[CSSPropertyWidth]

The CSS (CSS1) `width` property indicates the width of an element (excluding padding, borders and margins). It is overridden by the min-width or max-width properties, if either of them are present.

Valid property values (other than inherit and initial) are:

Value	Description
length	Width as a CSS length
%	Width of element defined as a percentage of width of its containing block
`auto`	(default value). Browser determines width

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.width="300px"`
Inherited: No
Animatable**:** Yes

word-break

[CSSPropertyWordBreak]

The CSS (CSS3) `word-break` property indicates the way in which words can be broken at line ends for scripts that are not Chinese, Japanese or Korean ("CJK").

Valid property values (other than inherit and initial) are:

Value	Description
`break-all`	Breaks can occur between any two letters
`keep-all`	Breaks are prohibited between pairs of letters
`normal`	(default value). Words break at line ends according to usual rules

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.wordBreak="keep-all"`
Inherited: Yes
Animatable: No

word-spacing

[CSSPropertyWordSpacing]

The CSS (CSS1) `word-spacing` property indicates the amount of whitespace between words.

Valid property values (other than inherit and initial) are:

Value	Description
length	Additional space between words as a CSS length, can be negative
`normal`	(default value). Normal space between words

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.wordSpacing="10px"`
Inherited: Yes
Animatable: Yes

word-wrap

[CSSPropertyWordWrap]

The CSS (CSS3) `word-wrap` property allows long words to be broken at line ends and to wrap onto the next line.

Valid property values (other than inherit and initial) are:

Value	Description
`break-word`	Otherwise unbreakable words can be broken
`normal`	(default value). Words can only be broken at allowed break points

Default Value: `normal`
JavaScript syntax: e.g. *object*`.style.wordWrap="break-word"`
Inherited: Yes
Animatable: No

z-index

[CSSPropertyZIndex]

The CSS (CSS1) `z-index` property specifies the stack order of an element, i.e. which is "in front of" other elements, and hence which is visible if several would otherwise appear in the same place. Elements with higher stack order (z-index value) are shown in preference to ones with lower stack order. It only works on positioned elements, i.e. with `position:absolute`, `position:relative` or `position:fixed`.

Valid property values (other than inherit and initial) are:

Value	Description
number	Stack order set to a specified value (which can be negative)
`auto`	(default value). Stack order of element set to the same as its parent's stack order

Default Value: `auto`
JavaScript syntax: e.g. *object*`.style.zIndex="-1"`
Inherited: No

CSS Lengths

[CSSLength]

Often it is important to specify the size of an HTML element. The conventions used when doing this in CSS are set out below.

A CSS length is a number followed by a length unit, such as 20px or 3cm. To be correctly understood, the specification needs to avoid any whitespace between the number and the length unit (e.g. using `20 px` will generally not be recognised as a length by browsers). If the value is zero (`0`) then the unit can be omitted.

Length units can be absolute or absolute.

Absolute lengths

These are fixed in size, and material will should appear exactly that size (unless the user then zooms in or out manually). As screen sizes vary considerably, best practice typically recommends using relative lengths not absolute lengths.

Unit	Description
`cm`	centimetres
`mm`	millimetres
`in`	inches (1in = 2.54cm)
`px`	pixels (1px = 1/96th of 1in, for high resolution screens, but for low resolution screens then 1px is one device pixel, i.e. dot, of the display
`pt`	points (1pt = 1/72 of 1in)
`pc`	picas (1pc = 12pt)

Relative lengths

Relative lengths are specified relative to another length property. These types of lengths tend to scale better across different screens or other rendering mediums.

Unit	Description
`ch`	Relative to width of a “0” (zero) character
`em`	Relative to font-size of element (e.g. 2em means twice the relevant font size)
`ex`	Relative to x-height of current font (this unit is rarely used)
`rem`	Relative to font-size of root element
`vw`	Relative to 1% of the width of the viewport (i.e. the browser window size)
`vh`	Relative to 1% of the height of the viewport (i.e. the browser window size)

`vmin`	Relative to 1% of the viewport's smaller dimension (not recognised by all browsers)
`vmax`	Relative to 1% of the viewport's larger dimension (not recognised by all browsers)

CSS Angles

[CSSAngle]

Some CSS property values are defined in terms of angles. It is used for example in rotate or skew parameters used by the transform property.

Angles can be defined in the following units:

Unit	Description
`deg`	Degrees. One full circle (360°) is `360deg`
`grad`	Gradians. One full circle (360°) is `400deg`
`rad`	Radians. One full circle (360°) is 2π radians, i.e. approximately `6.28318rad`
`turn`	Number of full turns. One full circle (360°) is `1turn`. The turn unit is not at the time of writing supported by all major browsers.

Positive angles represent clockwise turns, whilst negative angles represent counter-clockwise turns

Note: as with CSS lengths, no space should be left between the numerical value and unit. Unlike with CSS lengths, you always need to include a unit, i.e. 0 is not in itself a valid angle, even though `0deg` = `0grad` = `0rad` = `0turn`.

CSS Colours

[CSSColor]

A common activity within CSS (and any associated HTML markup) is to set the colour of an element. Colours can be specified in several ways:

- Using predefined names
- Using hexadecimal format
- Using RGB or RGBA formats
- Using HSL or HSLA formats

1. Predefined names

The following specific colours are listed in HTML / CSS specifications:

Colour name	Hexadecimal value	Colour
`AliceBlue`	`#F0F8FF`	
`AntiqueWhite`	`#FAEBD7`	
`Aqua`	`#00FFFF`	
`Aquamarine`	`#7FFFD4`	

Azure	#F0FFFF	
Beige	#F5F5DC	
Bisque	#FFE4C4	
Black	#000000	
BlanchedAlmond	#FFEBCD	
Blue	#0000FF	
BlueViolet	#8A2BE2	
Brown	#A52A2A	
BurlyWood	#DEB887	
CadetBlue	#5F9EA0	
Chartreuse	#7FFF00	
Chocolate	#D2691E	
Coral	#FF7F50	
CornflowerBlue	#6495ED	
Cornsilk	#FFF8DC	
Crimson	#DC143C	
Cyan	#00FFFF	
DarkBlue	#00008B	
DarkCyan	#008B8B	
DarkGoldenRod	#B8860B	
DarkGray	#A9A9A9	
DarkGrey	#A9A9A9	
DarkGreen	#006400	
DarkKhaki	#BDB76B	
DarkMagenta	#8B008B	
DarkOliveGreen	#556B2F	
DarkOrange	#FF8C00	
DarkOrchid	#9932CC	
DarkRed	#8B0000	
DarkSalmon	#E9967A	
DarkSeaGreen	#8FBC8F	
DarkSlateBlue	#483D8B	
DarkSlateGray	#2F4F4F	
DarkSlateGrey	#2F4F4F	
DarkTurquoise	#00CED1	
DarkViolet	#9400D3	
DeepPink	#FF1493	
DeepSkyBlue	#00BFFF	
DimGray	#696969	

DimGrey	#696969	
DodgerBlue	#1E90FF	
FireBrick	#B22222	
FloralWhite	#FFFAF0	
ForestGreen	#228B22	
Fuchsia	#FF00FF	
Gainsboro	#DCDCDC	
GhostWhite	#F8F8FF	
Gold	#FFD700	
GoldenRod	#DAA520	
Gray	#808080	
Grey	#808080	
Green	#008000	
GreenYellow	#ADFF2F	
HoneyDew	#F0FFF0	
HotPink	#FF69B4	
IndianRed	#CD5C5C	
Indigo	#4B0082	
Ivory	#FFFFF0	
Khaki	#F0E68C	
Lavender	#E6E6FA	
LavenderBlush	#FFF0F5	
LawnGreen	#7CFC00	
LemonChiffon	#FFFACD	
LightBlue	#ADD8E6	
LightCoral	#F08080	
LightCyan	#E0FFFF	
LightGoldenRodYellow	#FAFAD2	
LightGray	#D3D3D3	
LightGrey	#D3D3D3	
LightGreen	#90EE90	
LightPink	#FFB6C1	
LightSalmon	#FFA07A	
LightSeaGreen	#20B2AA	
LightSkyBlue	#87CEFA	
LightSlateGray	#778899	
LightSlateGrey	#778899	
LightSteelBlue	#B0C4DE	
LightYellow	#FFFFE0	

Lime	#00FF00	
LimeGreen	#32CD32	
Linen	#FAF0E6	
Magenta	#FF00FF	
Maroon	#800000	
MediumAquaMarine	#66CDAA	
MediumBlue	#0000CD	
MediumOrchid	#BA55D3	
MediumPurple	#9370DB	
MediumSeaGreen	#3CB371	
MediumSlateBlue	#7B68EE	
MediumSpringGreen	#00FA9A	
MediumTurquoise	#48D1CC	
MediumVioletRed	#C71585	
MidnightBlue	#191970	
MintCream	#F5FFFA	
MistyRose	#FFE4E1	
Moccasin	#FFE4B5	
NavajoWhite	#FFDEAD	
Navy	#000080	
OldLace	#FDF5E6	
Olive	#808000	
OliveDrab	#6B8E23	
Orange	#FFA500	
OrangeRed	#FF4500	
Orchid	#DA70D6	
PaleGoldenRod	#EEE8AA	
PaleGreen	#98FB98	
PaleTurquoise	#AFEEEE	
PaleVioletRed	#DB7093	
PapayaWhip	#FFEFD5	
PeachPuff	#FFDAB9	
Peru	#CD853F	
Pink	#FFC0CB	
Plum	#DDA0DD	
PowderBlue	#B0E0E6	
Purple	#800080	
RebeccaPurple	#663399	
Red	#FF0000	

RosyBrown	#BC8F8F	
RoyalBlue	#4169E1	
SaddleBrown	#8B4513	
Salmon	#FA8072	
SandyBrown	#F4A460	
SeaGreen	#2E8B57	
SeaShell	#FFF5EE	
Sienna	#A0522D	
Silver	#C0C0C0	
SkyBlue	#87CEEB	
SlateBlue	#6A5ACD	
SlateGray	#708090	
SlateGrey	#708090	
Snow	#FFFAFA	
SpringGreen	#00FF7F	
SteelBlue	#4682B4	
Tan	#D2B48C	
Teal	#008080	
Thistle	#D8BFD8	
Tomato	#FF6347	
Turquoise	#40E0D0	
Violet	#EE82EE	
Wheat	#F5DEB3	
White	#FFFFFF	
WhiteSmoke	#F5F5F5	
Yellow	#FFFF00	
YellowGreen	#9ACD32	

2. Hexadecimal format

All major browsers recognise hexadecimal format ('hex') colour values. These take the form `#RRGGBB` where the RR, GG and BB indicate the red, green and blue components of the colour in two-character hexadecimal format (i.e. ranging from 00 to FF, which in decimal correspond to 0 to 255 respectively. For example, `#FF0000` is red, since the red component is given its maximum value (FF, i.e. 255 in decimal format), whilst the green and blue components are given their minimum values, i.e. do not appear in the end colour. The hexadecimal format codes for each of the prespecified CSS colours are set out above.

3. RGB and RGBA formats

RGB colour values are also recognised by all major browsers. They are specified in the form `rgb`(*r*,*g*,*b*), where *r*, *g* and *b* are the red, green and blue components of the colour as above, but specified either in decimal values (0 to 255) or in percentage values (0% to 100%)

RGBA colour values are recognised by many major browsers. They extend the RGB to include an opacity (i.e. transparency) value. They are specified in the form `rgba`(*r*,*g*,*b*,*a*), where *r*, *g* and *b* are as above, and a (the 'alpha' parameter) can take a value between `0.0` (fully transparent, so invisible) and `1.0` (fully opaque, so will entirely block whatever is 'behind' the element assigned this opacity).

3. HSL and HSLA formats

HSL colour values are recognised by many browsers. HSL stands for hue, saturation and lightness. Such a colour is specified by `hsl`(*h*,*s*,*l*) where *h* refers to the hue, *s* the saturation and *l* the lightness of the colour.

HSL can be thought of as a cylindrical-coordinate representation of a colour. Hue defines the primary colour or mix (in degrees), between 0 and 360, with 0 (or 360) corresponding to red, 120 corresponding to green and 240 corresponding to blue (with intermediate values corresponding to mixtures of these colours). Saturation corresponds to the extent to which the colour diverges from grey, and is expressed in percentage terms, where 0% corresponds to grey and 100% to full colour. Lightness is also expressed in percentage terms, where 0% corresponds to black and 100% to white.

HSLA is like RGBA in extending the colour format to include an opacity value. It is specified by `hsla`(*h*,*s*,*l*,*a*), where *a* is the alpha parameter as above.

Appendix L: Default CSS Styles Applied to HTML Elements

[HTMLCSSDefaults]

The default CSS styles applied to different renderable HTML elements supported by HTML 5 are set out below.

Some element types have no applicable default CSS. These include:

HTML Element(s)	Default CSS applicable to that element
<abbr>, <audio>, <base>, <bdi>, , <button>, <canvas>, <data>, <dialog>, <input>, <keygen>, <main>, <menuitem>, <meta>, <meter>, <noscript>, <optgroup>, <option>, <progress>, <rp>, <ruby>, <select>, <source>, , <textarea>, <time>, <track>, <video>, <wbr>	None

For those that do have a default, occasionally this is browser specific, but usually it follows certain conventions:

HTML Element(s)	More	Default CSS applicable to that element
a:link	<a>	color: (*is an internal value specific to browser*); text-decoration: underline; cursor: auto;
a:visited	<a>	color: (*is an internal value specific to browser*); text-decoration: underline; cursor: auto;
a:link:active	<a>	color: (*is an internal value specific to browser*);
a:visited:active	<a>	color: (*is an internal value specific to browser*);
address	<address>	display: block; font-style: italic;
area	<area>	display: none;
article	<article>	display: block;
aside	<aside>	display: block;
b	<b>	font-weight: bold;
bdo	<bdo>	unicode-bidi: bidi-override;
blockquote	<blockquote>	display: block; margin: 1em 40px 1em 40px;
body	<body>	display: block; margin: 8px;
body:focus	<body>	outline: none;
caption	<caption>	display: table-caption; text-align: center;
cite	<cite>	font-style: italic;
code	<code>	font-family: monospace;
col	<col>	display: table-column;
colgroup	<colgroup>	display: table-column-group
datalist	<datalist>	display: none;
dd	<dd>	display: block; margin-left: 40px;

del	<del>	text-decoration: line-through;
details	<details>	display: block;
dfn	<dfn>	font-style: italic;
div		display: block;
dl	<dl>	display: block; margin: 1em 0 1em 0;
dt	<dt>	display: block;
em		font-style: italic;
embed:focus	<embed>	outline: none;
fieldset	<fieldset>	display: block; margin: 0 2px; padding: 0.35em 0.75em 0.625em; border: 2px groove (*and internal value*);
figcaption	<figcaption>	display: block;
figure	<figure>	display: block; margin: 1em 40px;
footer	<footer>	display: block;
form	<form>	display: block; margin-top: 0em;
h1	<h1>	display: block; font-size: 2em; margin: 0.67em 0; font-weight: bold;
h2	<h2>	display: block; font-size: 1.5em; margin: 0.83em 0; font-weight: bold;
h3	<h3>	display: block; font-size: 1.17em; margin: 1em 0; font-weight: bold;
h4	<h4>	display: block; margin: 1.33em 0; font-weight: bold;
h5	<h5>	display: block; font-size: 0.83em; margin-top: 1.67em 0; font-weight: bold;
h6	<h6>	display: block; font-size: 0.67em; margin-top: 2.33em 0; font-weight: bold;
header	<header>	display: block;
hr	<hr>	display: block; margin: 0.5em auto; border-style: inset; border-width: 1px;
html	<html>	display: block;
html:focus	<html>	outline: none;
i	<i>	font-style: italic;
iframe:focus	<iframe>	outline: none;
iframe[seamless]	<iframe>	display: block;
img	<img>	display: inline-block;

ins	<ins>	text-decoration: underline;
kbd	<kbd>	font-family: monospace;
label	<label>	cursor: default;
legend	<legend>	display: block; padding: 0 2px; border: none;
li		display: list-item;
link	<link>	display: none;
map	<map>	display: inline;
mark	<mark>	background-color: yellow; color: black;
menu	<menu>	display: block; list-style-type: disc; margin: 1em 0; padding-left: 40px;
nav	<nav>	display: block;
object:focus	<object>	outline: none;
ol		display: block; list-style-type: decimal; margin: 1em 0; padding-left: 40px;
output	<output>	display: inline;
p	<p>	display: block; margin: 1em 0;
param	<param>	display: none;
pre	<pre>	display: block; font-family: monospace; white-space: pre; margin: 1em 0;
q	<q>	display: inline;
q::before	<q>	content: open-quote;
q::after	<q>	content: close-quote;
rt	<rt>	line-height: normal;
s	<s>	text-decoration: line-through;
samp	<samp>	font-family: monospace;
script	<script>	display: none;
section	<section>	display: block;
small	<small>	font-size: smaller;
strong		font-weight: bold;
style	<style>	display: none;
sub	<sub>	vertical-align: sub; font-size: smaller;
summary	<summary>	display: block;
sup	<sup>	vertical-align: super; font-size: smaller;
table	<table>	display: table; border-collapse: separate; border-spacing: 2px; border-color: gray;
tbody	<tbody>	display: table-row-group; vertical-align: middle; border-color: inherit;

td	<td>	display: table-cell; vertical-align: inherit;
tfoot	<tfoot>	display: table-footer-group; vertical-align: middle; border-color: inherit;
th	<th>	display: table-cell; vertical-align: inherit; font-weight: bold; text-align: center;
thead	<thead>	display: table-header-group; vertical-align: middle; border-color: inherit;
title	<title>	display: none;
tr	<tr>	display: table-row; vertical-align: inherit; border-color: inherit;
u	<u>	text-decoration: underline;
ul		display: block; list-style-type: disc; margin: 1em 0; padding-left: 40px;
var	<var>	font-style: italic;

Appendix M: HTML Special Characters

[HTMLSpecialCharacters]

HTML markup includes some for special characters. The most frequently used are characters that if they appeared directly would be understood by HTML to relate to markup (see HTMLTutorialSpecialCharacters for further details).

Each of these markups is preceded by an ampersand, i.e. "`&`", followed by the markup name, followed by a semicolon, i.e. "`;`":

HTML name	character	name	Unicode hex code (decimal)
amp	&	ampersand	U+0026 (38)
quot	"	quotation mark	U+0022 (34)
apos	'	apostrophe	U+0027 (39)
lt	<	less than sign	U+003C (60)
gt	>	greater than sign	U+003E (62)
nbsp		non-breaking space	U+00A0 (160)
iexcl	¡	inverted exclamation mark	U+00A1 (161)
cent	¢	cent sign	U+00A2 (162)
pound	£	pound sign	U+00A3 (163)
curren	¤	currency sign	U+00A4 (164)
yen	¥	yen (yuan) sign	U+00A5 (165)
brvbar	¦	broken vertical bar	U+00A6 (166)
sect	§	section sign	U+00A7 (167)
uml	¨	diaeresis(or umlaut)	U+00A8 (168)
copy	©	copyright symbol	U+00A9 (169)
ordf	ª	feminine ordinal indicator	U+00AA (170)
laquo	«	left-pointing double angle quotation mark	U+00AB (171)
not	¬	not sign	U+00AC (172)
shy		soft (discretionary) hyphen	U+00AD (173)
reg	®	registered trademark symbol	U+00AE (174)
macr	¯	macron	U+00AF (175)
deg	°	degree symbol	U+00B0 (176)
plusmn	±	plus-minus sign	U+00B1 (177)
sup2	²	superscript two	U+00B2 (178)
sup3	³	superscript three	U+00B3 (179)
acute	´	acute accent	U+00B4 (180)
micro	µ	micro sign	U+00B5 (181)
para	¶	paragraph sign	U+00B6 (182)
middot	·	middle dot	U+00B7 (183)
cedil	¸	cedilla	U+00B8 (184)
sup1	¹	superscript one	U+00B9 (185)
ordm	º	masculine ordinal indicator	U+00BA (186)

raquo	»	right-pointing double angle quotation mark	U+00BB (187)
frac14	¼	fraction one quarter	U+00BC (188)
frac12	½	fraction one half	U+00BD (189)
frac34	¾	fraction three quarters	U+00BE (190)
iquest	¿	inverted question mark	U+00BF (191)
Agrave	À	Latin capital letter A with grave accent	U+00C0 (192)
Aacute	Á	Latin capital letter A with acute accent	U+00C1 (193)
Acirc	Â	Latin capital letter A with circumflex	U+00C2 (194)
Atilde	Ã	Latin capital letter A with tilde	U+00C3 (195)
Auml	Ä	Latin capital letter A with diaeresis	U+00C4 (196)
Aring	Å	Latin capital letter A with ring above	U+00C5 (197)
AElig	Æ	Latin capital letter AE	U+00C6 (198)
Ccedil	Ç	Latin capital letter C with cedilla	U+00C7 (199)
Egrave	È	Latin capital letter E with grave accent	U+00C8 (200)
Eacute	É	Latin capital letter E with acute accent	U+00C9 (201)
Ecirc	Ê	Latin capital letter E with circumflex	U+00CA (202)
Euml	Ë	Latin capital letter E with diaeresis	U+00CB (203)
Igrave	Ì	Latin capital letter I with grave accent	U+00CC (204)
Iacute	Í	Latin capital letter I with acute accent	U+00CD (205)
Icirc	Î	Latin capital letter I with circumflex	U+00CE (206)
Iuml	Ï	Latin capital letter I with diaeresis	U+00CF (207)
ETH	Ð	Latin capital letter Eth	U+00D0 (208)
Ntilde	Ñ	Latin capital letter N with tilde	U+00D1 (209)
Ograve	Ò	Latin capital letter O with grave accent	U+00D2 (210)
Oacute	Ó	Latin capital letter O with acute accent	U+00D3 (211)
Ocirc	Ó	Latin capital letter O with circumflex	U+00D4 (212)
Otilde	Õ	Latin capital letter O with tilde	U+00D5 (213)
Ouml	Ö	Latin capital letter O with diaeresis	U+00D6 (214)
times	×	multiplication sign	U+00D7 (215)
Oslash	Ø	Latin capital letter O with stroke (Latin capital letter O slash)	U+00D8 (216)
Ugrave	Ù	Latin capital letter U with grave accent	U+00D9 (217)
Uacute	Ú	Latin capital letter U with acute accent	U+00DA (218)
Ucirc	Û	Latin capital letter U with circumflex	U+00DB (219)
Uuml	Ü	Latin capital letter U with diaeresis	U+00DC (220)
Yacute	Ý	Latin capital letter Y with acute accent	U+00DD (221)
THORN	Þ	Latin capital letter THORN	U+00DE (222)
szlig	ß	Latin small letter sharp s, i.e. German Eszett	U+00DF (223)
agrave	à	Latin small letter a with grave accent	U+00E0 (224)
aacute	á	Latin small letter a with acute accent	U+00E1 (225)
acirc	â	Latin small letter a with circumflex	U+00E2 (226)
atilde	ã	Latin small letter a with tilde	U+00E3 (227)

auml	ã	Latin small letter a with diaeresis	U+00E4 (228)
aring	å	Latin small letter a with ring above	U+00E5 (229)
aelig	æ	Latin small letter ae	U+00E6 (230)
ccedil	ç	Latin small letter c with cedilla	U+00E7 (231)
egrave	è	Latin small letter e with grave accent	U+00E8 (232)
eacute	é	Latin small letter e with acute accent	U+00E9 (233)
ecirc	ê	Latin small letter e with circumflex	U+00EA (234)
euml	ë	Latin small letter e with diaeresis	U+00EB (235)
igrave	ì	Latin small letter i with grave accent	U+00EC (236)
iacute	í	Latin small letter i with acute accent	U+00ED (237)
icirc	î	Latin small letter i with circumflex	U+00EE (238)
iuml	ï	Latin small letter i with diaeresis	U+00EF (239)
eth	ð	Latin small letter eth	U+00F0 (240)
ntilde	ñ	Latin small letter n with tilde	U+00F1 (241)
ograve	ò	Latin small letter o with grave accent	U+00F2 (242)
oacute	ó	Latin small letter o with acute accent	U+00F3 (243)
ocirc	ô	Latin small letter o with circumflex	U+00F4 (244)
otilde	õ	Latin small letter o with tilde	U+00F5 (245)
ouml	ö	Latin small letter o with diaeresis	U+00F6 (246)
divide	÷	division sign (obelus)	U+00F7 (247)
oslash	ø	Latin small letter o with stroke (Latin small letter o slash)	U+00F8 (248)
ugrave	ù	Latin small letter u with grave accent	U+00F9 (249)
uacute	ú	Latin small letter u with acute accent	U+00FA (250)
ucirc	û	Latin small letter u with circumflex	U+00FB (251)
uuml	ü	Latin small letter u with diaeresis	U+00FC (252)
yacute	ý	Latin small letter y with acute accent	U+00FD (253)
thorn	þ	Latin small letter thorn	U+00FE (254)
yuml	ÿ	Latin small letter y with diaeresis	U+00FF (255)
OElig	Œ	Latin capital ligature oe	U+0152 (338)
oelig	œ	Latin small ligature oe	U+0153 (339)
Scaron	Š	Latin capital letter s with caron	U+0160 (352)
scaron	š	Latin small letter s with caron	U+0161 (353)
Yuml	Ÿ	Latin capital letter y with diaeresis	U+0178 (376)
fnof	ƒ	Latin small letter f with hook (function, florin)	U+0192 (402)
circ	ˆ	modifier letter circumflex accent	U+02C6 (710)
tilde	˜	small tilde	U+02DC (732)
Alpha	A	Greek capital letter Alpha	U+0391 (913)
Beta	B	Greek capital letter Beta	U+0392 (914)
Gamma	Γ	Greek capital letter Gamma	U+0393 (915)
Delta	Δ	Greek capital letter Delta	U+0394 (916)

Epsilon	Ε	Greek capital letter Epsilon	U+0395 (917)
Zeta	Ζ	Greek capital letter Zeta	U+0396 (918)
Eta	Η	Greek capital letter Eta	U+0397 (919)
Theta	Θ	Greek capital letter Theta	U+0398 (920)
Iota	Ι	Greek capital letter Iota	U+0399 (921)
Kappa	Κ	Greek capital letter Kappa	U+039A (922)
Lambda	Λ	Greek capital letter Lambda	U+039B (923)
Mu	Μ	Greek capital letter Mu	U+039C (924)
Nu	Ν	Greek capital letter Nu	U+039D (925)
Xi	Ξ	Greek capital letter Xi	U+039E (926)
Omicron	Ο	Greek capital letter Omicron	U+039F (927)
Pi	Π	Greek capital letter Pi	U+03A0 (928)
Rho	Ρ	Greek capital letter Rho	U+03A1 (929)
Sigma	Σ	Greek capital letter Sigma	U+03A3 (931)
Tau	Τ	Greek capital letter Tau	U+03A4 (932)
Upsilon	Υ	Greek capital letter Upsilon	U+03A5 (933)
Phi	Φ	Greek capital letter Phi	U+03A6 (934)
Chi	Χ	Greek capital letter Chi	U+03A7 (935)
Psi	Ψ	Greek capital letter Psi	U+03A8 (936)
Omega	Ω	Greek capital letter Omega	U+03A9 (937)
alpha	α	Greek small letter alpha	U+03B1 (945)
beta	β	Greek small letter beta	U+03B2 (946)
gamma	γ	Greek small letter gamma	U+03B3 (947)
delta	δ	Greek small letter delta	U+03B4 (948)
epsilon	ε	Greek small letter epsilon	U+03B5 (949)
zeta	ζ	Greek small letter zeta	U+03B6 (950)
eta	η	Greek small letter eta	U+03B7 (951)
theta	θ	Greek small letter theta	U+03B8 (952)
iota	ι	Greek small letter iota	U+03B9 (953)
kappa	κ	Greek small letter kappa	U+03BA (954)
lambda	λ	Greek small letter lambda	U+03BB (955)
mu	μ	Greek small letter mu	U+03BC (956)
nu	ν	Greek small letter nu	U+03BD (957)
xi	ξ	Greek small letter xi	U+03BE (958)
omicron	ο	Greek small letter omicron	U+03BF (959)
pi	π	Greek small letter pi	U+03C0 (960)
rho	ρ	Greek small letter rho	U+03C1 (961)
sigmaf	ς	Greek small letter final sigma	U+03C2 (962)
sigma	σ	Greek small letter sigma	U+03C3 (963)
tau	τ	Greek small letter tau	U+03C4 (964)
upsilon	υ	Greek small letter upsilon	U+03C5 (965)
phi	φ	Greek small letter phi	U+03C6 (966)

chi	χ	Greek small letter chi	U+03C7 (967)
psi	ψ	Greek small letter psi	U+03C8 (968)
omega	ω	Greek small letter omega	U+03C9 (969)
thetasym	ϑ	Greek theta symbol	U+03D1 (977)
upsih	ϒ	Greek Upsilon with hook symbol	U+03D2 (978)
piv	ϖ	Greek pi symbol	U+03D6 (982)
ensp		en space[d]	U+2002 (8194)
emsp		em space[d]	U+2003 (8195)
thinsp		thin space[d]	U+2009 (8201)
zwnj		zero-width non-joiner	U+200C (8204)
zwj		zero-width joiner	U+200D (8205)
lrm		left-to-right mark	U+200E (8206)
rlm		right-to-left mark	U+200F (8207)
ndash	–	en dash	U+2013 (8211)
mdash	—	em dash	U+2014 (8212)
lsquo	‘	left single quotation mark	U+2018 (8216)
rsquo	’	right single quotation mark	U+2019 (8217)
sbquo	‚	single low-9 quotation mark	U+201A (8218)
ldquo	“	left double quotation mark	U+201C (8220)
rdquo	”	right double quotation mark	U+201D (8221)
bdquo	„	double low-9 quotation mark	U+201E (8222)
dagger	†	dagger, obelisk	U+2020 (8224)
Dagger	‡	double dagger, double obelisk	U+2021 (8225)
bull	•	bullet (black small circle)	U+2022 (8226)
hellip	…	horizontal ellipsis (three dot leader)	U+2026 (8230)
permil	‰	per mille sign	U+2030 (8240)
prime	′	prime (minutes, feet)	U+2032 (8242)
Prime	″	double prime (seconds, inches)	U+2033 (8243)
lsaquo	‹	single left-pointing angle quotation mark	U+2039 (8249)
rsaquo	›	single right-pointing angle quotation mark	U+203A (8250)
oline	‾	overline (spacing overscore)	U+203E (8254)
frasl	⁄	fraction slash (solidus)	U+2044 (8260)
euro	€	euro sign	U+20AC (8364)
image	ℑ	black-letter capital I (imaginary part)	U+2111 (8465)
weierp	℘	script capital P (power set, Weierstrass p)	U+2118 (8472)
real	ℜ	black-letter capital R (real part symbol)	U+211C (8476)
trade	™	trademark symbol	U+2122 (8482)
alefsym	ℵ	alef symbol (first transfinite cardinal)	U+2135 (8501)
larr	←	leftwards arrow	U+2190 (8592)
uarr	↑	upwards arrow	U+2191 (8593)
rarr	→	rightwards arrow	U+2192 (8594)
darr	↓	downwards arrow	U+2193 (8595)

harr	↔	left right arrow	U+2194 (8596)
crarr	↵	downwards arrow with corner leftwards (carriage return)	U+21B5 (8629)
lArr	⇐	leftwards double arrow	U+21D0 (8656)
uArr	⇑	upwards double arrow	U+21D1 (8657)
rArr	⇒	rightwards double arrow	U+21D2 (8658)
dArr	⇓	downwards double arrow	U+21D3 (8659)
hArr	⇔	left right double arrow	U+21D4 (8660)
forall	∀	for all	U+2200 (8704)
part	∂	partial differential	U+2202 (8706)
exist	∃	there exists	U+2203 (8707)
empty	∅	empty set (null set)	U+2205 (8709)
nabla	∇	del or nabla (vector differential operator)	U+2207 (8711)
isin	∈	element of	U+2208 (8712)
notin	∉	not an element of	U+2209 (8713)
ni	∋	contains as member	U+220B (8715)
prod	∏	n-ary product (product sign)	U+220F (8719)
sum	∑	n-ary summation	U+2211 (8721)
minus	−	minus sign	U+2212 (8722)
lowast	∗	asterisk operator	U+2217 (8727)
radic	√	square root (radical sign)	U+221A (8730)
prop	∝	proportional to	U+221D (8733)
infin	∞	infinity	U+221E (8734)
ang	∠	angle	U+2220 (8736)
and	∧	logical and (wedge)	U+2227 (8743)
or	∨	logical or (vee)	U+2228 (8744)
cap	∩	intersection (cap)	U+2229 (8745)
cup	∪	union (cup)	U+222A (8746)
int	∫	integral	U+222B (8747)
there4	∴	therefore sign	U+2234 (8756)
sim	∼	tilde operator (varies with, similar to)	U+223C (8764)
cong	≅	congruent to	U+2245 (8773)
asymp	≈	almost equal to (asymptotic to)	U+2248 (8776)
ne	≠	not equal to	U+2260 (8800)
equiv	≡	identical to or 'equivalent to'	U+2261 (8801)
le	≤	less-than or equal to	U+2264 (8804)
ge	≥	greater-than or equal to	U+2265 (8805)
sub	⊂	subset of	U+2282 (8834)
sup	⊃	superset of	U+2283 (8835)
nsub	⊄	not a subset of	U+2284 (8836)
sube	⊆	subset of or equal to	U+2286 (8838)
supe	⊇	superset of or equal to	U+2287 (8839)

`oplus`	⊕	circled plus (direct sum)	U+2295 (8853)
`otimes`	⊗	circled times (vector product)	U+2297 (8855)
`perp`	⊥	up tack (orthogonal to, perpendicular)	U+22A5 (8869)
`sdot`	⋅	dot operator	U+22C5 (8901)
`lceil`	⌈	left ceiling	U+2308 (8968)
`rceil`	⌉	right ceiling	U+2309 (8969)
`lfloor`	⌊	left floor	U+230A (8970)
`rfloor`	⌋	right floor	U+230B (8971)
`lang`	⟨	left-pointing angle bracket (bra)	U+2329 (9001)
`rang`	⟩	right-pointing angle bracket (ket)	U+232A (9002)
`loz`	◊	lozenge	U+25CA (9674)
`spades`	♠	spade suit	U+2660 (9824)
`clubs`	♣	club suit (shamrock)	U+2663 (9827)
`hearts`	♥	heart suit (valentine)	U+2665 (9829)
`diams`	♦	diamond suit	U+2666 (9830)

Note: an ampersand "`&`" will usually display if it is not part of a special character, but it is better to use HTML markup for it, i.e. `&`.