

---

# HTML5 (BY DAVID KELLEHER)

## SEPARATION OF CONCERNS

Computer code should be written for one task at a time. Coding each individual task in a different file or function is called *Separation of Concerns*.

A block of code should not try to accomplish more than one task at a time, like defining both the visual style and the function of an interface button. If different people are coding different tasks, and modify the same code, they are more likely to interfere with each other's work.

Also, the task performed by a block of code should not also be accomplished elsewhere, because then a programmer must search many locations to find all the instances of relevant code.

### **Separation of Concerns for Front-End Web Developers**

- HTML (Hypertext Markup Language) = **Content** (semantics)
- CSS (Cascading Style Sheets) = **Design** (type, color, layout)
- JS (JavaScript) = **Features** (interactivity)

## SEMANTICS

Semantics is the study of meaning.

When writing HTML, tags are used to identify the meaning of each part of a document. Parts of a document include articles, titles, paragraphs, emphasized text, links, images, and footers.

The browser needs to know what each part of a document is to render the text properly on the screen. Search engines also take advantage of HTML semantics, using clues such as headings to identify the topic of a page.

### **Which parts of a document are identified in HTML?**

- Any elements that help a search engine or automated program interpret the contents of a document
- Any elements where a developer will want to apply design styles to change how they look in the browser
- Any elements that need to be made available for user interactions, like links, buttons, and hotspots
- Any elements relevant to assistive technologies, like text spoken in a specific manner by a screen reader

HTML is designed to be simple, and therefore limits the number of parts which must be identified.

### **ALWAYS Avoid design decisions in HTML code**

Adding HTML tags to a document changes how it looks, because web browsers include a built-in stylesheet with default design styles. And a primary reason for writing HTML code is to identify blocks of text that need to be styled a certain way. However, to follow the separation of concerns principle, coders should avoid choosing specific HTML elements only to achieve a certain look in the browser. HTML elements should always be selected for semantically correct reasons.

## HTML SYNTAX

An **element** is a part of a web document. Elements usually include a start tag, content, and end tag.

```
<p lang="en">This is text tagged using the paragraph element.</p>
```

A **start tag** is an element name surrounded by angled brackets.

```
<p>
```

An **attribute** modifies elements, is placed in the start tag, and has a value inside quotes.

```
lang="en"
```

An **end tag** repeats the start tag, with a leading slash.

```
</p>
```

A **void** element has no content, and no end tag. An example is the image element `<img>`.

## ELEMENT CONTENTS

**Text** is the most common type of content in an HTML document, including letter, number, punctuation, and symbol characters.

**Character references** in text begin with an ampersand and end with a semicolon. The code in the middle could be a name, decimal number, or hexadecimal number.

`&amp;` (displays the ‘&’ ampersand character)

References can be used to encode special Unicode characters, like the copyright symbol © and Microsoft Word curly quotes. However, modern browsers support those special characters, so encoding Unicode characters is no longer necessary. The one reason to use character references is to escape characters in code.

**Escaped characters** solve the problem that arises when you want to display text characters that could be interpreted as HTML code. For example, `<p>` is normally interpreted by browsers as the start of a paragraph. If you wanted to display `<p>` as text content in the browser instead, use the character references: `&lt;p&gt`

Display Character	Use this instead	Where should you use it?
&	<code>&amp;amp;</code>	Anywhere in document
<	<code>&amp;lt;</code>	Anywhere in document
>	<code>&amp;gt;</code>	Anywhere in document
'	<code>&amp;apos;</code>	In attribute values: <code>&lt;option value="O&amp;apos;Leary"&gt;</code>
"	<code>&amp;quot;</code>	In attribute values: <code>&lt;option value"&amp;quot;A&amp;quot;"&gt;</code>

**Comments** help coders document their work. Documentation is important to help other web page authors understand how and why a site was coded. Comments are ignored by browsers and can only be viewed in the source code. Comments begin with the string "<!--" and end with the string "-->"

```
<!-- This is a comment -->
```

The **svg** element defines a format for representing vector graphics in code. SVG graphics can scale to any size, unlike PNG, JPG, and GIF images. There are practical challenges to using svg graphics, so embedded images are almost always used instead.

The **math** element allows use of the MathML specification. MathML is a markup language that describes mathematical formulas so they can be displayed in web pages. Unfortunately, it is not well supported in browsers. JavaScript formula parsers are used instead.

**Nested Elements** are elements that are embedded inside other elements. They must be closed in reverse order. In the following example, the paragraph element is opened before the strong element, so it must be closed after the strong element.

```
<p>The sign said <strong>Do Not Enter</strong></p>
```

## VALID HTML5 CODE

A **validator** like the one at <https://validator.w3.org/> can be used to check if syntax is correct.

Most errors are caused by simple typos or incorrect punctuation. Here are some common mistakes:

- Unexpected Markup – the element name does not exist in the specs or is used where it is not allowed.
- Unclosed Elements – required closing tags are missing, or typed incorrectly (for example, <p/>)
- Misnested Tags – Elements are not closed in reverse order.
- Improper Attributes – the attribute name does not exist in the specs.
- Improper Attribute Values – values with spaces are not surrounded by quotation marks, or are missing a quotation mark, or include a character that must be escaped.

## HTML5 CODE STYLE

The rules for writing HTML5 are very flexible. The creators of the HTML5 language assumed the average person won't learn a lot of technical rules.

- Browsers ignore extra whitespace characters, and multiple spaces are reduced to one on the screen.
- Most element and attributes names can be written in either uppercase or lowercase letters.
- Attribute values can be written with single quotes, double quotes, and sometimes no quotes at all
- The closing tag is optional for many elements.

The result can be valid markup that looks like a complete mess:

```
<BLOCKQUOTE>
<p LANG= en> This is a <b>blockquote &#x26; paragraph
</b>
demo</p></Blockquote>
```

Since unorganized, inconsistent code can be hard to read and maintain, most recommend following a style guide with consistent rules for writing code. Different guides have different rules. Here is a common style guide:

- Block-level elements (sections that begin on a new line in the browser) start on a new line in the code
- Nested block-level elements should be indented.
- All code including HTML5 element and attribute names should be lowercase.
- Attribute values should use double quotes.
- Optional closing tags should not be omitted.
- Character entity references shouldn't be used unless they are escaping the HTML characters noted above.

```
<blockquote>
  <p lang="en">This is a <b>blockquote &#x26; paragraph</b> demo</p>
</blockquote>
```

A code styling tool can be used to quickly fix formatting and style issues in code. A popular tool is <https://dirtymarkup.com/>

## BOILERPLATE

Boilerplate is a skeleton of HTML code, including markup that should appear on every page in a website. It speeds up the creation of new web page templates. The description and author meta tags are not required by the specs but are recommended. Here is a boilerplate example, with bolded text identifying what the author of the page needs to edit:

```
<!doctype html>

<html lang="en">
<head>
  <meta charset="utf-8"/>

  <title>Page Title</title>
  <meta name="description" content="This is the Page Description.">
  <meta name="author" content="Firstname Lastname">
</head>

<body>
  ... page content ...
</body>
</html>
```

## DOCTYPE

The **Document Type Declaration** (DOCTYPE) associates a webpage with a Document Type Definition (DTD), which defines the set of markup tags that can be used on the page. It is case sensitive.

- The HTML5 DOCTYPE should be the first line of HTML on each page.

```
<!doctype html>
```

## ROOT ELEMENT

**html** is the root element, which means it is the first and top node in an HTML document.

- This element is optional, although most style guides will say to include it.
- Adding the language attribute is also recommended for all HTML5 documents

```
<html lang="en">
  ...
</html>
```

## DOCUMENT METADATA

Metadata is information that describes the content of web pages and is never shown to the user.

**head** is the first element inside an html element and is required. It is a collection of metadata about the page, including the page title, stylesheet links, page styles, and meta tags.

```
<head>
  ...
</head>
```

**title** is a **required** element inside the head element. The title can be found in the browser's title bar or tabs, user history, bookmarks, and searches.

```
<title>Page Title</title>
```

**link** connects the HTML document to other resources. The most common usage is to attach a stylesheet.

```
<link rel="stylesheet" href="default.css">
```

**style** is another element used for adding styles to a document. It overrides the stylesheet but should be avoided to follow the separation of concerns principle.

```
<style>
  body { color: blue; background: white; }
</style>
```

**script** is an element that connects the HTML document to an external code library. In the header, it is considered metadata.

```
<script src="functions.js"></script>
```

**meta** is a generic metadata element used when information cannot be expressed by any of the other previously listed metadata elements.

A character set should be defined for every HTML document. That is most easily accomplished with a meta element. Use UTF-8, which supports symbols and characters from other languages that do not use the Latin alphabet, such as Hebrew and Chinese. Not defining the charset could cause text to appear garbled on a page. That frequently happens with Microsoft Word special characters like curly quotes.

```
<meta charset="utf-8"/>
```

There are standard metadata names defined in HTML. A useful one is *description*, which is shown in search engine results and affects click-through rates. The *author* is not commonly used but can help an organization keep track of who created each page. The once popular *keyword* has little value now since it was been used by many people to spam search engines.

```
<meta name="description" content="David's training site teaches HTML5.">
<meta name="author" content="David Kelleher">
```

## DOCUMENT OUTLINE

The structure of a document can be represented as a **document outline**, which is essentially like the table of contents in a report. For example:

1. Heading Introduction
2. Main Navigation Menu
3. About Me
  - a. Biography
  - b. Filmography
  - c. Discography

Document outlines are defined using sectioning elements in the HTML.

## SECTION CONTENT

Every page has a body element wrapped around all page content. New sectioning elements introduced in HTML5 (article, nav, aside, section, header, and footer) are optional, but should be used when the group's significance is enough to warrant an entry in the page's document outline.

**body** is the 2<sup>nd</sup> element inside the html element, after <head>, and is placed right after the head element is closed. All page content displayed in the browser window goes inside the body element.

```
<body>
  ...
</body>
```

**nav** is an element used to group navigation links. It is only used for **major navigation blocks** such as the **main and secondary menus** on a page, or **breadcrumb navigation**. nav should not be used for miscellaneous links in the page footer or lists of links to other resources. Navigation items are usually marked up as lists.

Because nav is also a sectioning element, it contributes to the document's outline.

```
<nav>
  ...
</nav>
```

**article** is an element representing a complete self-contained block of content. That means the block of content would be complete and make sense on its own if distributed in syndication or reused independently. Articles could be used for the following types of sections: **forum post, blog entry, news article, user-submitted comment, interactive widget**, or other **independent item of content**.

Because article is also a sectioning element, it contributes to the document's outline.

```
<article>
  ...
</article>
```

**aside** is an element representing content that is tangentially related to content around the aside. Potential uses for aside include *pull quotes*, *sidebars*, *advertisements*, and *groups of nav elements*.

Because aside is also a sectioning element, it contributes to the document's outline.

The ARIA (Accessible Rich Internet Application) specification helps make web page content more accessible to people with disabilities. Additional attributes can be added to HTML elements to help the disabled users better navigate and understand content on a page. For example, an author should add the search role for an aside containing an interactive search form.

```
<aside>
  ...
</aside>
```

```
<aside role="search">
  ...
</aside>
```

**section** is used for any other thematic grouping of content, typically with a heading. It should not be used when a more specific sectioning element such as nav, article, or aside would be appropriate. It should also not be used when elements are grouped together only for style or scripting purpose; use <div> instead.

Because section is a sectioning element, it contributes to the document's outline.

```
<section>
  ...
</section>
```

**h1, h2, h3, h4, h5, h6** "heading" elements represent headings for their sections. They are used to briefly describe the topic of the sections they introduce. The h1 element has the highest rank and h6 has the lowest rank.

The first heading in a section defines the section name in the document outline. Sectioning elements are optional where headings are used. Additional headings result in the start of new implied sections when they have equal rank (i.e. <h2> and <h2>), or new implied subsections when the additional headings have lower rank (i.e. <h1> and <h2>).

```
<h1>Cats</h1>
  <h2>Persian Cats</h2>
  ...
  <h2>Siamese Cats</h2>
  ...
<h1>Dogs</h1>
  <h2>Chihuahua Dogs</h2>
  ...
  <h2>Beagle Dogs</h2>
```

Resulting Document Outline:

1. Cats
  - a. Persian Cats
  - b. Siamese Cats
2. Dogs
  - a. Chihuahua Dogs
  - b. Beagle Dogs



While the above example is valid HTML, authors could also wrap all headings inside their own explicit section elements, instead of using heading elements alone to create implied sections:

```
<article>
  <h1>Cats</h1>
  <section>
    <h2>Persian Cats</h2>
    ...
  </section>
  <section>
    <h2>Siamese Cats</h2>
    ...
  </section>
</article>
<article>
  <h1>Dogs</h1>
  <section>
    <h2>Chihuahua Dogs</h2>
    ...
  </section>
  <section>
    <h2>Beagle Dogs</h2>
    ...
  </section>
</article>
```

Resulting Document Outline (it is the same):

1. Cats
  - a. Persian Cats
  - b. Siamese Cats
2. Dogs
  - a. Chihuahua Dogs
  - b. Beagle Dogs

The only type of content allowed inside heading elements is **phrasing content**. Phrasing content is defined in the HTML5 specs and includes text-level semantics, most embedded content, links, and some form controls.

**header** elements represent a group of introductory content or navigational aids for its parent's section. It is not needed when the header only contains a single heading like h1. It is commonly used to group multiple elements in a section's header, such as the **heading, byline, logos, search form, social networking links**, and other **navigation shortcuts**.

The header is not a sectioning element; it doesn't introduce a new section or appear in a document's outline.

```
<header>
  ...
</header>
```

Here is a header with a heading, a subheading with a span element for different styling, and a byline:

```
<header>
  <h1>
    Heading
    <span>Tagline or Subtitle</span>
  </h1>
  Byline: David Kelleher
</header>
```

**footer** elements include metadata about the content in its parent's section. It is separated from the main content of the section. Examples of content found in footers include the *copyright*, *fine print*, *appendices*, and *indexes*. The footer is normally placed at the bottom of the section but isn't required to go there. The footer could include content that could also go in the header, such as the author and navigation links.

The footer is not a sectioning element; it doesn't introduce a new section or appear in a document's outline.

```
<footer>  
  ...  
</footer>
```

**address** is an element representing the contact information for its parent's section. It could contain the *author's web page*, *email address*, or *postal address*. Typically, the address is put in the footer. The address element should not be used to markup other addresses within the content that are not the section's contact information. The address element should also not include information other than contact information.

The address is not a sectioning element; it doesn't introduce a new section or appear in a document's outline.

```
<address>  
  ...  
</address>
```

Here is an example of an HTML template with commonly used section elements:

```
<!DOCTYPE html>

<html lang="en">
<head>
  <meta charset="utf-8"/>

  <title>Page Title</title>
  <meta name="description" content="This is the Page Description.">
  <meta name="author" content="Firstname Lastname">
</head>

<body>
  <header>
    <h1>Page Header</h1>
    ... other page header content ...
  </header>

  <nav>
    ... main navigation content ...
  </nav>

  <article>
    <h2>Article Header</h2>
    ... article content ...
    <aside>
      <h3>Sidebar Header</h3>
      ... article sidebar content ...
    </aside>

    <footer>
      <address>
        ...address content ...
      </address>
      ... other footer content ...
    </footer>
  </article>

  <aside>
    <h2>Related Content Header</h2>
    ... related content ...
  </aside>

  <footer>
    ... page footer content ...
  </footer>

</body>
</html>
```



**blockquote** is an element that represents a section quoted from another source.

Short quotes are usually done in-line with quotation marks. The **blockquote** is used for longer quotes or any quote where separation from the surrounding content groups is desired. The **cite** attribute links to the source; it must contain a valid URL. Attribution for the quote must be placed outside the **blockquote** element.

```
<blockquote>
  <p> Don't cry because it's over, smile because it happened.</p>
</blockquote>
```

The **cite** attribute can be used to mark up the name of the source in conjunction with the **blockquote** element.


```
<p>His next piece was the aptly named <cite>Sonnet 130</cite>:</p>
<blockquote cite="http://quotes.example.org/s/sonnet130.html">
  <p>My mistress' eyes are nothing like the sun,<br>
  Coral is far more red, than her lips red</p>
</blockquote>
```

**ul**, **ol**, **li** elements are used to create lists.

Unordered lists are created with the **ul** element and are usually styled with bullet points.

List items are created with the **li** element, and nested inside unordered and ordered lists. List items can contain any type of content, including sections and nested lists. However, it is not a good idea to place heading tags directly in list items, because that splits the list into multiple sections, which doesn't make sense semantically. Note that the bullet points and ordered list numbers are not included in the content text.

```
<ul>
  <li>Norway
  <li>Switzerland
  <li>United Kingdom
</ul>
```




Ordered lists are created with the **ol** element and are usually styled with numbers and sometimes letters, defined by the **type** attribute. Ordered lists also can have a **start** attribute to determine the first list item's value (always defined as a decimal number), and the **reversed** attribute to count backwards.

Ordered list types:

1	decimals (1. 2. 3.) ← This is the default
a	lowercase (a. b. c.)
A	uppercase (A. B. C.)
i	lowercase roman numerals (i. ii. iii.)
I	uppercase roman numerals (I. II. III.)

```
<ol type="I" start="3" reversed>
  <li>May
  <li>April
  <li>March
</ol>
```



**dl**, **dt**, **dd** elements are used to create description lists. Those are name-value groups, with the names tagged using **dt** and the values tagged using **dd**. For each group, one or more **dt** name elements are followed by one or more **dd** value element.

Description lists may be used to markup terms and definitions, metadata topics and values, questions and answers, or any other groups of name-value data pairs. Other examples include **glossaries** and **FAQs** (frequently asked questions.)

Any content except section content may be included in **dt** elements. There are no content restrictions for **dd**.

```
<dl>
  <dt> Last modified time </dt>
  <dd> 2004-12-23 </dd>
  <dt> Authors </dt>
  <dt> Editors </dt>
  <dd> Robert Rothman </dd>
  <dd> Daniel Jackson </dd>
</dl>
```

Last modified time 2004-12-23
Authors
Editors
Robert Rothman
Daniel Jackson

**figure** elements represent self-contained content referenced from the main content.

**figcaption** elements are optional captions for figures.

The **figure** element is meant to be used when the content is directly related to the flow of the main content. The **aside** section element should be used instead when content is only tangentially related or doesn't illustrate the content in the surrounding flow. The **blockquote** element should be used when the content couldn't be moved outside its position in the main content flow.

Depending on context, the following items are appropriate for figures: **photographs**, **videos**, **code examples**, **data tables**, and **illustrative quotes**.

```
<figure>
  <p>'Twas brillig, and the slithy toves<br>
  Did gyre and gimble in the wabe;<br>
  All mimsy were the borogoves,<br>
  And the mome raths outgrabe.</p>
  <figcaption><cite>Jabberwocky</cite> Lewis Carroll</figcaption>
</figure>
```

**main** element can be used to group the content in a page unique to that document. It should not include site logos, branding, main navigation, copyright, search forms, and other elements reused across multiple pages. Only one **main** element is allowed per page. Website visitors looking at a web page in a browser can visually locate the main content of a page with ease. However, blind screen reader users, search engines, automated data miners, and browsers on small devices benefit from having a coded way to skip directly to main content.

```
<main>
  ...
</main>
```

**div** was a frequently used element in HTML 4. It is used for breaking pages into content blocks for layout creation, styling, and script access. However, it has no semantic meaning, so HTML5 documents should use the section and grouping elements that have specific semantic meanings whenever possible. Use `<div>` as a last resort where no other element has an appropriate semantic meaning for the contents.

The `div` element can be targeted by JavaScript and stylesheets.

```
<div>  
    ...  
</div>
```

## TEXT-LEVEL SEMANTICS

**Text-level semantics defines sections of inline content. An example would be a few words highlighted within a sentence.**

The **strong** attribute represents increased importance (the contents matter more), seriousness (the contents contain a warning or caution), or urgency (the contents should be seen first). Strong importance implies there will be consequences when ignored. It displays as bold text but shouldn't be used just to make text look bold.

'strong' text should be emphasized when read or spoken by a screen reader, though due to lack of consistency among authors screen readers do not.

```
<strong>Warning!</strong>. This dungeon is dangerous.
```

The **b** element represents a span of text to which attention is being drawn without conveying any extra importance. It should be used as a last resort to make text stylistically different from its surroundings, when no other element that conveys more specific semantic information is appropriate. Use the **b** element for:

- Key words in a document
- Product names in a review
- Actionable words in interactive text-driven software

'b' text should **not** be emphasized when spoken by a screen reader.

By default, it displays as bold text. The class attribute could be used to specify the reason why the element is being used for styling purposes. For example:

```
The <b class="keyword">strong</b> attribute is used for urgent warnings.  
The <b class="product">oreo cookie</b> was tasty.  
You enter a small room. Your <b class="object">sword</b> glows brighter.
```

The **em** attribute represents stress emphasis. By default, it displays as italic text, but shouldn't be used just to italicize text.

'em' text should be emphasized when spoken by a screen reader, though due to lack of consistency among authors screen readers do not.

```
It was the <em>first</em> time I felt appreciated.  
It was the first time <em>I</em> felt appreciated.  
It was the first time I felt <em>appreciated.</em>
```



The **i** element represents a span of text in an alternate voice or mood, or is otherwise offset from the normal prose in a manner indicating a different quality of text. Examples include:

- Taxonomic designation
- Technical term
- Idiomatic phrase from another language
- Transliteration
- A thought
- Ship name in Western texts

'I' text should **not** be emphasized when spoken by a screen reader.

By default, it displays as italic text, but should not just be used to make text look italic. The class attribute could be used to specify the reason why the element is being used.

```
The <i class="taxonomy">Felis silvestris catus</i> is cute.  
<i class="term">Conditioned taste aversion</i> is why he didn't eat it.  
There is a certain <i lang="fr">je ne sais quoi</i> in the air.  
I rode the French <i class="transliteration" lang="fr">Métro</i>  
Raymond slept. <i>The ship sailed away on Thursday</i>, he dreamt.  
These are the voyages of the <i>Starship Enterprise</i>.
```

The **dfn** element represents the defining instance of a term. It usually displays as italic text.

```
The <dfn>dfn</dfn> element is used to identify terms to be defined.
```

The **abbr** element represents an abbreviation or acronym. The title attribute may be specified to expand the abbreviation. It usually displays as dotted underlined text. It can be combined with the dfn element.

```
<abbr>WHATWG</abbr>
```

```
<dfn>Web Hypertext Application Technology Working Group</dfn>  
(<abbr title="Web Hypertext Application Technology Working  
Group">WHATWG</abbr>)  
is a loose unofficial collaboration of Web browser manufacturers.
```

The **cite** element represents the title of a work like a book. It is not meant to be used for quotes and complete citations, but it should be used for the title of the work within the citation. By default, it displays as italic text.

```
<cite>Universal Declaration of Human Rights</cite>, United Nations,  
December 1948. Adopted by General Assembly resolution 217 A (III).
```

The **q** element represents content quoted from another source. It usually renders as fancy “curly quotes.” Because special characters like curly quotes can now be embedded in HTML documents with UTF-8 encoding, there are few cases where it makes sense to use it anymore. One case is when the optional cite attribute is used to link to the quote’s source.

```
<p>The W3C document <cite>About W3C</cite> says the W3C's mission is <q cite="http://www.w3.org/Consortium/">To lead the World Wide Web to its full potential by developing protocols and guidelines that ensure long-term growth for the Web</q>.
```

The **mark** element represents a run of text marked or highlighted for reference purpose, due to its relevance in another context. It could be used to emphasize words for exam study that were not emphasized in the original text, or to highlight search terms on a search results page.

```
<p lang="en-US">Consider the following quote:</p>
<blockquote lang="en-GB">
  <p>Look around and you will find, no-one's really
  <mark>colour</mark> blind.</p>
</blockquote>
<p lang="en-US">As we can tell from the <em>spelling</em> of the word, the person writing this quote is clearly not American.</p>
```

The **sup** element represents a superscript and the **sub** element represents a subscript.

```
<abbr>M<sup>lle</sup></abbr> Gwendoline
```

```
<var>x<sub>10</sub></var>, <var>y<sub>10</sub></var>
```

The **small** element is used for fine print.

```
<small>© 2019 David Kelleher</small>
```

The **s** element represents contents that are no longer accurate or relevant. By default, it displays as strikethrough text.

```
Retail price <s>$5.99</s> now just $4.99!
```

The **br** void element represents a line break. It should not be used to start new paragraphs or add whitespace. It must be used only for line breaks that are part of the content, as in poems or addresses.

```
<p>P. Sherman<br>
42 Wallaby Way<br>
Sydney</p>
```

The **wbr** void element represents a line break opportunity. It could be used to suggest line break points for very long strings of text without whitespace, or for long lines of computer code.

```
Supercalifragilistic<wbr>  
expialidocious
```

The **time** element is used to add a machine-readable copy of the date and/or time contents. The **datetime** attribute is optional, but there isn't a reason to use the `<time>` element if the `datetime` is not included

```
<time datetime="2005-10-05">October 5</time>
```

The **data** element includes a machine-readable form of its contents in the required **value** attribute. For example, the months of the year could be represented by numbers.

```
Order the new <data value="X850D">2019 Sony BRAVIA</data> television.
```

The **u** element represents content with non-textual annotation. It usually displays as squiggly underlined text. The specs only recommend using this element for Chinese proper name marks and words labeled as being misspelt.

```
This word is <u>misspelled</u> and marked as such in the text editor.
```

Several elements exist for the purpose of marking up computer code. The **code** element represents a fragment of computer code and displays in a monospaced font. The **kbd** element represents user input (such as from a keyboard) and the **samp** element represents computer output. They also display in a monospaced font. The **var** element represents a variable in a mathematic expression or computer code, or a symbol identifying a physical quantity. It usually displays as italic text.

The **ruby**, **rp**, **rt** elements are used to mark content with ruby annotations. Ruby annotations are short runs of text presented alongside base text, primarily used in East Asian typography as a guide for pronunciation. Other elements used for internationalization are **bdi** and **bdo**, which are used to handle text written in right-to-left or unknown directions.

The **span** element has no semantic meaning, but could be used with the `class`, `lang`, or `dir` elements for purposes such as adding styles to small runs of text. It is used as a last resort when no other text-level semantic element is appropriate.

```
<span> ... </span>
```

## LINKS

The **a** element is used to create hyperlinks. It is a text-level element. In HTML5, this element can now be wrapped around any other elements, including headings and large sections.

The **href** attribute is required to define a link and should be the path to access the new file.

```
<a href="about.html" target="_blank">About Us</a>
```

The href attribute can be skipped to create a **placeholder** link that can't be clicked.

```
<a>placeholder example</a>
```

To open the link in a new browser tab or window, specify the **target** attribute with the value `_blank`.

```
<a href="http://www.desera.com/" target="_blank">Desera</a>
```

The **download** attribute can specify that a link to a file, like an mp3 or video, should be downloaded instead of opened in the browser. Unfortunately, browser support is limited, and the attribute will only work for about half of all website visitors.

The **rel** attribute describes the relationship of the linked document to the current document. It is most often used to identify files that are external stylesheets. More than a dozen link types are defined in the specs for use with the rel attribute.

author	link to a page about the author of the current content
bookmark	url representing the permalink to the current content
help	link to context sensitive help
license	link to a page with the copyright license content
next	link to the next page or document in a series
nofollow	the linked content is not endorsed (blocks search engines)
noreferrer	referrer headers should not be sent
prefetch	the target resource should be preemptively cached
prev	link to the previous page or document in a series
search	link to a feature to search the current page or site
tag	Gives a tag that applies to the current document

```
Written by<a href="about.html" rel="author">David Kelleher</a>
```

The **fragment identifier** is added to a URL with the hash tag and is used to link to part of a page. When clicked, the browser will jump to the element that has an ID value matching the fragment identifier.

```
<a href="faq.html#question5">Direct link to question 5</a>
```

The **query string** is added to a URL with the question mark and is used to send information to the server when clicked.

```
<a href="product.html?id=3&color=red">View product #3 in red.</a>
```

**Absolute links** target a specific resource anywhere on the web. They must include `http://` and the complete domain name, followed by an optional path and page name.

```
<a href="http://www.desera.com/">
```

**Relative links** target another resource within the current website. Paths are created using the following rules:

- If the target resource is in the same folder, just the page name can be specified.
- If the resource is in a subfolder, the path to the resource must be included, including slashes to navigate into each folder.
- If the resource is in a parent folder, two dots and a slash can be used to move up a folder level.

```
<a href="folder1/folder2/childpage.html">  
<a href=" ../parentpage.html">
```

## EMBEDDED CONTENT

Embedded content allows pictures, videos, nested HTML frames, and plugin content to be added to a web page. Authors should use the height and width attributes to provide hints to the browser, so it can render the page layout more quickly. *The height and width attributes should not be used to resize images.*

The **img** element represents an image. The required **src** attribute should be the url of the image resource. The **alt** attribute should provide alternative content for those who cannot see or process images.

```

```

The **iframe** element represents a nested browsing context. That means it is a web page embedded in a web page frame. It is most commonly used to embed videos and widgets from third party sites like YouTube.

```
<iframe width="560" height="315" src="https://www.youtube.com/  
embed/kiyi-C7NQRQ" ... </iframe>
```

In HTML5, there are new **video** and **audio** elements. Upload mp4 videos and mp3 audio files. Alternate text content should be provided for accessibility.

```
<video controls autoplay>  
  <source src='video.mp4' type='video/mp4'>  
</video>
```

The video and audio elements are frequently enhanced using JavaScript and CSS. Those who want feature rich players on their own sites, without embedding from services like YouTube, could develop on using JavaScript and CSS. Or, existing JavaScript libraries like JW Player and JPlayer could be used.

The **area** element is used to create client-side image maps with hyperlinked hot spots.

## TABULAR DATA

Tables are data with more than one dimension, like a spreadsheet or other content arranged in rows and columns.

The **table** element represents tabular data. It supports the *sortable* attribute, which hasn't been implemented yet but, in the future, will allow tables to be sorted by selected columns.

The **tr** element represents a row of cells in a table.

The **td** element represents a single cell in a table. Therefore, a tr element will contain one or more td elements. More complex table structures can be created using the *colspan*, *rowspan*, and *headers* attributes.

```
<table>
  <tr>
    <td> A Grade </td>
    <td> B Grade </td>
  </tr>

  <tr>
    <td> 90-99 </td>
    <td> 80-89 </td>
  </tr>
</table>
```

A Grade	B Grade
90-99	80-89

The optional **th** element represents a single header cell in a table. If header cells are desired, they are usually placed in the first row and first column of the table. They are usually displayed using bold text. More complex table structures can be created using the *colspan*, *rowspan*, *headers*, *scope*, and *abbr* attributes.

A table can start with a **caption** element, which describes the content of the table. The table can then have optional **colgroup** and child **col** elements to define column groupings. Next, row groupings can be defined using an optional **thead** element, one or more **tbody** elements, and a **tfoot** element immediately after the thead or following all tbody and tr elements.

```
<table>
  <caption>Positive and Negative Characteristics</caption>
  <thead>
    <tr>
      <th> Characteristic </th>
      <th> Negative </th>
      <th> Positive </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <th> Mood </th>
      <td> Sad </td>
      <td> Happy </td>
    </tr>
    <tr>
      <th> Grade </th>
      <td> Failing </td>
      <td> Passign </td>
    </tr>
  </tbody>
</table>
```

Positive and Negative Characteristics

<b>Characteristic</b>	<b>Negative</b>	<b>Positive</b>
<b>Mood</b>	Sad	Happy
<b>Grade</b>	C	A+

## FORMS

Forms are components of web pages containing controls such as text fields, checkboxes, color pickers, and buttons. Submitting a form communicates with the server, and passes user submitted data for additional processing on the server. HTML5 created many new input element types and attributes, but due to limited support, Javascript fallbacks are strongly recommended. Developers can also add client and server-side validation code to check user input.

The **form** element represents a form. It usually includes the **method**, **action**, and **enctype** attributes to specify how and where form data will be submitted to the server. The new **autocomplete** attribute specifies regardless of whether the form's fields can be autofilled by the browser. The new **novalidate** feature will stop the browser from enforcing client-side validation. The **name** attribute is used when a developer wants to reference the form to write custom JavaScript validation code.

```
<form action="submit.php" method="post" enctype="multipart/form-data">
    ...
</form>
```

The **input** element is a data field usually with a form control (like a text box) that allows users to edit the data.

```
<form action="submit.php" method="post" enctype="multipart/form-data">
    <input type="text" name="fullname">
</form>
```

Other form elements are **textarea** for text boxes, **radio** for radio buttons, **checkbox** for checkboxes, and **submit** for buttons. The **checked** attribute will select a default value for the user.

```
<form action="serverpage.php" method="post" enctype="multipart/form-
data">
    Message: <br/>
    <textarea name="msg"></textarea><br/>

    <p>Customer:</p>
    <input type="radio" name="customer" value="new">New
    <input type="radio" name="customer" value="returning">Returning

    <input type="checkbox" name="notify" value="updates" checked>
    I'd like to receive notices when this website updates.
    <input type="checkbox" name="notify" value="services">
    I'd like to receive notices pertaining to related websites.

    <input type="submit" value="Submit">
</form>
```



In HTML5, new form elements were added. Support in browsers is still inconsistent and incomplete and require JavaScript fallbacks. They include search, email, url, tel, number, range, date, month, week, time, datetime, datetime-local, color

For more information on HTML5 form elements see:  
<http://html5doctor.com/html5-forms-input-types/>

The **fieldset** element groups a set of controls in the form. They can be nested within other fieldsets. The **legend** element provides a caption for a fieldset.

```
<fieldset>
  <legend>User Data</legend>
  Name: ...
  Email: ...
</fieldset>
```

The **label** element associates a caption with a specific form control. If both the caption and form control are not next to each other in the code, and can't be wrapped with a label element, the **for** and **id** attributes can be used to associate the caption with the control. The label element should always be used. Including the for attribute improves accessibility because the technique is widely implemented in assistive technologies to help blind users figure out which label goes with each control.

```
<label>Full Name <input name="fullname"></label>
```

```
<label for="fullname">Full Name</label> <input name="fullname"
id="fullname"></label>
```

## GLOBAL ATTRIBUTES

**Global Attributes** are attributes that can be used with any element.

The **style** attribute adds CSS styles to the element. This should be avoided in order to follow the separation of concerns principle.

The **id** attribute provides a hook for custom scripts and styles. Each id must be unique within a page.

The **class** attribute also labels specific elements. Class values can be reused throughout the page.

Several global attributes enhance **accessibility** by making it easier to navigate a page with the keyboard, including **accesskey** and **tabindex**. However, these are not commonly used because browsers, screen readers, and other assistive technologies have already defined most keys for other shortcuts.

A few global attributes assist with **internationalization**, which is support for multiple languages. The **lang** attribute specifies the content's language and should be used with the html element on every page. The **translate** attribute is used to identify which content on a page should or should not be translated when the page is localized (translated into other languages for foreign users). The **dir** attribute specifies the directionality of the element's text, which defaults to left-to-right for English but is right-to-left for other languages.

The following example indicates the paragraph is written in Spanish:

```
<p lang="es">
```

The list of valid languages can be found here:

<http://www.iana.org/assignments/language-subtag-registry/language-subtag-registry>

Several attributes were added for **interactive** application development purposes. The **contextmenu** attribute defines popup right-click context menus. The **draggable** and **dropzone** attributes are used for creating drag and drop functionality. The **title** attribute values usually display as popup tooltips but are not viewable on touch devices so its usefulness is limited. The **hidden** and **inert** attributes affect whether elements are seen on the screen and whether they can be interacted with using the mouse and keyboard.

Two attributes for editing, which happens in areas such as text input boxes on a form, are **contenteditable** and **spellcheck**.

COPYRIGHT NOTICE

THIS DOCUMENT INCLUDES EXAMPLES QUOTED FROM THE PUBLIC W3C HTML5 SPECIFICATIONS

OTHER CONTENT COPYRIGHT 2019 DAVID KELLEHER